Chapter

ARCHITECTURE AND DESIGN METHODOLOGY FOR HIGHLY-RELIABLE TSV-NOC SYSTEMS

***Khanh N. Dang*[1], *and Abderazek Ben Abdallah*[2]**
E-mail address: benab@u-aizu.ac.jp.
[1]SISLAB, The University of Engineering and Technology,
Vietnam National University Hanoi, Hanoi, Vietnam
[2]Adaptive Systems Laboratory,
Graduate School of Computer Science and Engineering,
University of Aizu, Aizu-Wakamatsu, Fukushima, Japan

Abstract

During the past few decades, a lot of research has been focusing on Three-dimensional Networks-on-Chips (3D-NoCs) as an auspicious solution to alleviate the interconnect bottleneck and reduce the power consumption in current System-on-Chips (SoCs) designs. However, 3D-NoC systems are becoming susceptible to a variety of faults caused by crosstalk, radiation, oxide breakdown, and so on. As a result, a simple failure in a single transistor caused by one of these factors may compromise the entire system reliability where the failure can be illustrated in corrupted message delivery, time requirement unsatisfactory, or even sometimes the whole system collapse. This chapter presents a detailed faults/defects analysis and an efficient reliability assessment method to approximate the lifetime reliability of a NoC system. Also, this chapter presents an architecture and hardware design of a fault-tolerant TSV based 3D-NoC system which can handle major failures (i.e., hard-faults, soft-errors and TSV-defects) that can occur in TSV-based 3D-NoC systems.

Keywords: Fault-tolerance, TSV-NoC, Analytical Model, Design

1. Introduction

3D-ICs have been proposed to address the domination of wire-delay over gate-delay [1]. By porting a given system to the third dimension, the global wire-length can be shortened thanks to the use of Through-Silicon-Vias (TSVs) [2]. As a result, the total power consumption can be reduced and the performance can be further enhanced. Meanwhile, Network-on-Chip (NoC)

This is a preprint of a book chapter in Horizons in Computer Science Research, Volume 16, Chapter 7, pp 199-246. URL: `https://www.novapublishers.com/catalog/product_info.php?products_id=64154`. ISBN: 978-1-53613-327-1. Copyright (c) 2018 Nova Science Publishers.

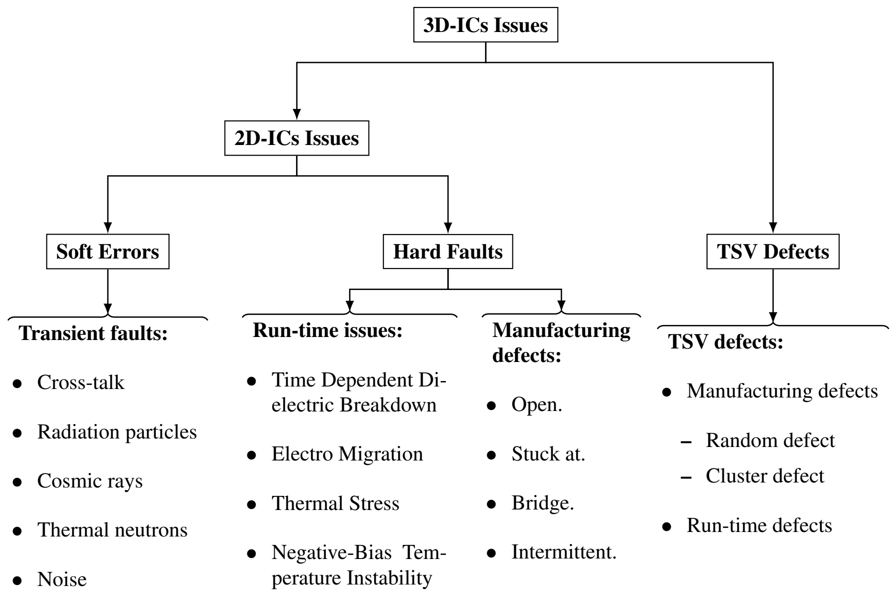


Figure 1. Taxonomy of reliability issues in 3D-ICs.

has been widely considered as one of the most promising replacements for traditional communication paradigms (e.g., bus, point-to-point) by offering better scalability and parallelism [3]. The fusion form of these two advanced technologies is known as 3D-Network-on-Chips (3D-NoCs) [4,5]. By combining the benefits of 3D-ICs and NoC paradigms, 3D-NoCs have opened a new horizon for high performance and low power designs which have become crucial to satisfy the strict requirements of future complex applications [6,7].

While 3D-NoCs have been increasing in popularity, they are threatened by the decreasing reliability of aggressively scaled transistors and the high defect-rates of TSVs. In fact, transistors are approaching the fundamental limits of scaling, and gate widths are nearing the molecular scale; thus, resulting in breakdowns and wear-outs in end products [8,9]. Moreover, the anticipated fabrication geometry in 2018 scales down to $8nm$ with projected 0.6V supply voltage [10]. Since, the low supply voltage enforces a very narrow noise margin, architectures have become more vulnerable and sensitive to faults. As shown in the Fig. 1, faults in 3D-NoCs can be classified into three types:

- *Hard faults*, including both permanent and intermittent faults, can occur during the manufacturing stage or under specific operation circumstances. Although intermittent faults can disappear after a specific period, their inconsistency can be treated as permanent faults to avoid complex situations.

- *Soft errors* do not permanently defect the gate and only occur over a short period. Because of their special characteristics, they are unpredictable and unavoidable.

- *TSV defects* are a direct consequence of the low yield rates of 3D-ICs due to the imperfection of the manufacturing process [11,12]. The TSV defect-rates have been reported as

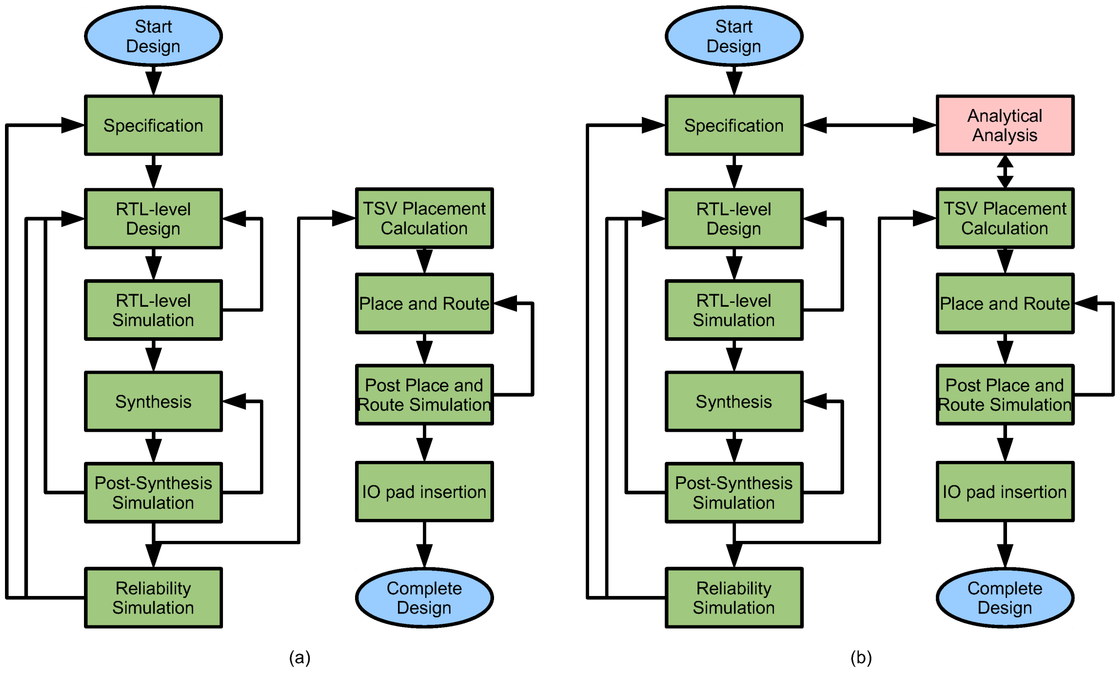


Figure 2. Design methodology: (a) Conventional method; (b) Proposed.

nearly 0.63% [13]. As a result, TSVs in 3D-NoCs have become more fault sensitive, not only in the manufacturing phase; but, also during the operation time.

As each of the above faults has different impacts on the system's performance and reliability, they cannot be treated similarly when designing highly reliable 3D-NoCs. Nevertheless, none the studies conducted so far addressed the system reliability when the above three types of faults are present in the system at the same time. All the proposed approaches deal with one, or two at maximum, type of faults. In this fashion, we cannot provide enough reliability when it comes to practical cases. In fact, when running a manufactured 3D-NoC system, there is a considerable chance that the three fault's types occur at the same time. In this case, the system may fail if there are no fault-tolerant mechanisms that deal with all types of faults.

Another consequence of the complicated nature of 3D-NoC systems is the design complexity and how the amount of time and effort dedicated to consider reliability can be a critical challenge. Although there is a substantial number of fault-tolerance works for NoCs, which were summarized in [14], selecting the appropriate method can be an important design choice. Moreover, several works have different configurations (e.g., number and level of modular redundancies [15], checkpoint/recovery interval [16]) and design trade-offs (additional power consumption, performance degradation, extra area cost) which create difficulties of selecting/designing the most suitable fault-tolerant technique for a given system. Therefore, their reliability efficiency need to be carefully investigated before choosing the appropriate method. This is in contrast to many of the previously proposed fault-tolerant systems where, as depicted in Fig. 2 (a), the reliability assessment is done in later design stages, such as gate-level and post-layout simulations.

In this chapter, we propose a complete architecture, called TSV-FETO, and a comprehensive design methodology for highly-reliable 3D-NoC systems, as presented in Fig. 2 (b). In contrast to previously conducted research, the proposed approach not only presents several mechanisms and algorithms that are capable of dealing with hard-faults, soft-errors and TSV-defects; but, it also relies on a low-overhead analytic model.

- A scalable TSV utilization architecture and methodology to tackle the lack of reliability in inter-layer links. To reducing the TSV-cluster defects, a router corrects its defected TSV cluster by choosing one of its four neighbor TSV-clusters located on the same layer.
- A light-weight analytical platform to evaluate fault-tolerant NoC architectures. Based on the proposed approach, we measure the fault-tolerance efficiency using a new parameter, named Reliability Acceleration Factor (RAF). The goal of this model is to provide an efficient and accurate reliability assessment to help designers easily understand and evaluate the advantages and drawbacks of their potential fault-tolerance methods.
- To deal with hard-faults and soft-errors, the proposed systems rely on the efficient mechanisms and algorithms that we previously proposed in [17, 18]. Using these techniques, TSV-FETO is capable of detecting and recovering from soft errors which occur in the routing pipeline stages, in addition to its capabilities to handle permanent faults in links, input buffers, and crossbars.

The organization of this chapter is as follows: in Section 2, we present the related works on fault-tolerance and reliability assessment. Section 3 presents the proposed reliability assessment methodology. Section 4 presents the TSV-FETO - the hard fault, soft error, and TSV defect tolerant 3D-NoC architecture. In Section 5, we describe the TSV fault-tolerant architecture and algorithms. Section 6 provides the implementation and evaluation results. Finally, we present the conclusion and future work in the last section.

2. Related Literatures

In this section, we provide a brief survey on the 3D-NoCs fault-tolerance techniques. More detailed surveys about fault-tolerant works can be found in our published works [17–19] or in a NoC fault-tolerance survey [14]. Here, we highlight the methods and existing works in Table 1. Since this chapter focuses on the design methodology of highly reliable 3D-NoCs where reliability assessment and TSV fault-tolerance play important roles, this section will discuss in details about these two focuses.

2.1. Fault Tolerance Methodologies

Depend on the type of faults, designers may have different approaches. As we previously mentioned, faults in 3D-NoCs can be classified into three types where they are differently addressed.

Hard faults are usually addressed by replacements or self-configuration to adapt to the occurrences of faults. [29,30] use spare wires to recover faulty ones. In [15,18] authors use redundant

Table 1. Taxonomy of the different error recovery protocols and architectures in NoCs.

Fault Type	Position/Method	Fault Tolerant Method
Soft Errors	Data Path	Automatic Re-transmission Request [20]
		Error Detecting/Correcting Code [21,22]
	Control Logic	Logic/Latch Hardening [14,23]
		Pipeline Redundancy [24]
		Monitoring and Correcting model [25–28]
Hard Faults	Routing Technique	Spare wire [29,30]
		Split transmission [31]
		Fault-Tolerant routing algorithm [32,33]
	Architecture-based Technique	Hardware Redundancy [15,34,35]
		Reconfiguration architectures [33–36]
TSV Defects	Redundancy	Shifting [13,37]
		Crossbar [38]
		Network [39]
	Management	Design awareness [40,41]
		Randomly distributed redundancy [11]

modules to handle the task without changing the overall performance. Since using redundancies demands extra modules, self-configuration is more cost-effective. For instance, using fault-tolerant routing algorithms [32,33] can help the network work around the fault without the need for redundancies. However, this type of techniques may create bottle-necks inside the system which degrades the performance.

The soft error recovery techniques consist of two main approaches: software and hardware. In the software approach, checkpoints are created and the system roles back to the checkpoints whenever soft error crash it [24]. On the other hand, the hardware approach normally focuses on protecting the fundamental devices against soft errors. For instance: error correcting code [21, 22] or logic/latch hardening [14,23] can be applied to correct soft errors.

The TSV fault-tolerance can be considered as hard fault-tolerance due to the characteristic of TSVs. Here, we classified them into three layer *Physical layer*, *Data-link layer* and *System layer*. In *Physical layer*, the improvement of TSV manufacturing/designing can help to reduce the defect-rate [40,42]. Some TSV faults can be corrected using a dedicated circuit [41]. In the *Data-link layer*, the most common method is adding redundant TSVs to correct the defected ones [38,39,43]. In [11], the authors propose a mapping method to reduce the impact of cluster defects. As vertical wires, defected TSVs can also be detected and recovered by *Error Correction Coding* [21]. In the *System layer*, which mainly focuses on 3D-NoCs, fault-tolerant routing algorithms [32] are one of the most suitable solutions.

One of the matters that are still under investigation is the TSV failure distribution. In general, there are two main assumptions for the failure distribution: *Random* [38,44] and *Clustering* distributions [11,37,39]. *Random* TSV defect is efficiently dealt by adding redundancy and recovery methods; but, *Clustering* defects still remain as a considerable challenge [19]. Moreover, TSV misalignment [38] also may occur and is classified as a cluster defect. Because of the stress and thermal issues, the TSVs may also be defected after manufacturing. In [45], the authors presented several *Mean Time To Failure* equations of 3D-ICs when affected by *Time Dependent Dielectric Breakdown*, *Thermal Cycling* and *Electro-migration* where the temperature values play an important role. Because of the clustering effect on hot-spot areas in 3D-ICs [46], the obvious result was found to be the TSV-cluster defect.

As shown in Table 1, a numerous amount of solutions have been efficiently dealt with several types of faults. However, non of them have completely tackled all soft errors, hard faults, and TSV defects. In fact, all the mentioned fault may occur, a completed solution is needed.

2.2. Reliability Assessment

Table 2. Reliability assessment methodologies.

Method	Description	Type
Physical Failure Analysis [47]	The acceleration factors of the failure rate are defined as Oxide, Metalization, Hot Carrier, Contamination, Package, EOS/ESD (Electrical Overstress/ Electrostatic Discharge) and Miscellaneous Failure Rate.	physical-level
Soft Error Rate Estimation [48]	This simulation flow uses a random function to generate hit locations and perform analysis using HSPICE simulation. This aims to obtain the probability of soft errors with several physical parameters.	physical-level
Chip-level Soft Error Estimation [49]	The authors present a method to estimate the failure of a full-chip. It is based on timing analyses of basic devices. A combination of these analyses helps build the final full-chip failure rate.	physical-level and system-level simulation
Monte-Carlo Simulation [50,51]	A random error injection and function verification to measure the reliability of the system under failure.	system-level simulation
System Reliability Block Diagrams [50]	This model presents the logic relationship of the system's components. There are five basic systems: serial, parallel, stand-by, k-out-of-n and complex systems. All of these systems are analyzed using the probability theory.	analytical model
Fault-tree Analysis [50]	A graphical and logical representation of possible events occurring in a system.	analytical model
Markov Model [50, 52]	It models the system's possible failure situations as a set of states. There are possible failure rates and repair rates between two states. The final reliability of the system is obtained by calculating the probability of the dead-end states.	analytical model
TSV Failure Analysis [45]	This analytical model analyzes the probability of failure of 3D-NoCs under the failure of TSVs. In 3D-NoCs, TSVs are used as the vertical connection wires.	analytical model (for NoCs)
NoCs Reliability Analysis [53]	This model estimates the reliability of a NoC system using probability theory.	analytical model (for NoCs)

Table 2 shows the different methodologies used to analyze a given system's reliability. We categorized reliability assessment into three basic methods: *Physical-level analysis*, *System-level simulation*, and *Analytical models*. The *Physical-level analysis* [47,48] [54] can obtain accurate results for gates or small devices; however, it is time-consuming and requires a huge computation resources to analyze complex systems. The *System-level simulation* can be performed by a Monte-Carlo simulation [50] or a RTL-level simulation [14,21,55,56]. In order to evaluate the system's reliability, the *Mean Time To Failure* [57] estimation can be used. Beside analysis, design-awareness is one of important keys to obtain reliable systems [58–61]. The *Analytical models* [9,45,62,63] can be used to save a significant time wasted on redesign. The IEEE 1413.1 [50] recommends several methods: *System Reliability Block Diagrams*, *Fault-tree*

Analysis, and *Markov-model* for repairable systems. Analytical models for NoCs are recently presented [64–67]. A reliability assessment for TSV failure in 3D-NoCs is addressed in [45,68]. The reliability of a NoC is also analyzed in [53, 69]. These methods have provided promising solutions for NoCs' reliability assessment; however, they lack the support for fault-tolerant and highly complex systems.

3. Reliability Assessment Method

This section presents the basic concepts, definitions, and assumptions used in the proposed analytical assessment methodology. We firstly start by giving an overview of the Markov-state model followed by the different assumptions and models adopted in the proposed assessment method. Then, we provide a quantitative reliability assessment method for Network-on-Chips.

3.1. Markov-State Model and Assessment Definitions

3.1.1. Markov State Model Overview

A system operating with faults can be converted into a Markov state model [52, 70] as shown in Fig. 3. Each state S_i of the Markov model represents a possible *status* (event, configuration, behavior) of the system where a *status* can be a case where one or multiple elements of the system fail. Depending on the ability to operate of the system, a state is classified as "healthy" or "faulty". The transitions between states are represented with one of two values: λ - fault-rate of a component; and μ - repair rate of a component.

The reliability of a system can be defined as a time-dependent probability function $R(t)$ in the time domain ($R^*(s)$ in *Laplace* domain) and can be evaluated using the *Mean Time To Failure* (MTTF) [52] calculation as follows:

$$\mathbf{MTTF} = \int_{t=0}^{\infty} R(t) = \lim_{s\to 0}(R^*(s)) = \lim_{s\to 0}(\sum_{S_i\in\mathbb{H}} P(S_i)) \tag{1}$$

Where $\mathbb{H}$ is the set of healthy states.

3.1.2. Assumptions

In a fault-tolerant system, the fault-tolerant method has to improve the reliability of the system. As a result, the MTTF value has to be also increased. Therefore, we propose the *Reliability Acceleration Factor* (RAF) to denote the efficiency of fault tolerance, represented as:

$$\mathsf{RAF} = \frac{\lambda_{original}}{\lambda_{FT}} = \frac{\mathbf{MTTF}_{FT}}{\mathbf{MTTF}_{original}} \geq 1 \tag{2}$$

Where:

- $\mathbf{MTTF}_{original}$ is the Mean Time To Failure of the original system.
- $\mathbf{MTTF}_{FT}$ is the Mean Time To Failure of the fault-tolerant system.

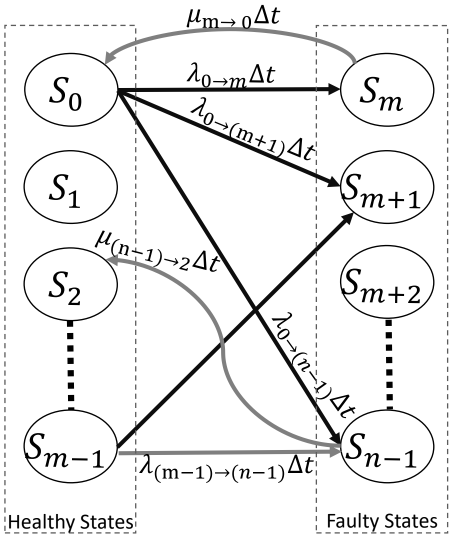

$\mu_{m\to 0}\Delta t$
S_0
S_1
S_2
S_{m-1}
S_m
S_{m+1}
S_{m+2}
S_{n-1}
$\lambda_{0\to m}\Delta t$
$\lambda_{0\to(m+1)}\Delta t$
$\mu_{(n-1)\to 2}\Delta t$
$\lambda_{0\to(n-1)}\Delta t$
$\lambda_{(m-1)\to(n-1)}\Delta t$
Healthy States
Faulty States

Figure 3. A Markov-state reliability model for an n states system with m non-faulty states.

The transitions have dedicated "steady-state" rates which need to be predefined [52]. Here, we also assume the "raw" fault-rate of a system with k components as follows:

$$\lambda_{system} = \frac{1}{\mathbf{MTTF}_{system}} = \sum_{i=1}^{k} f_i \pi_i \lambda_{unit} \tag{3}$$

where $unit$ is a selected module as a reference for calculation. π_i is the fault-rate ratio between the component and the *unit*. It can be defined as the area cost ratio and operating time ratio [47,62]. f_i is the fault-rate ratio given by a fault-tolerant technique ($f = 1/RAF$).

3.1.3. Classified Model

We categorize fault-tolerance architectures [14,62,71] into four basic models where each model is treated separately and differently: *Non-fault tolerant model*, *Spare model*, *Fault-reduced model*, and *Error handling model* as:

- *Model 0 - Non-fault tolerant model*: This model is applied for the module without fault-tolerance capabilities. Its fault-rate can be obtained by Equation 3 or based on physical-level analysis.
- *Model 1 - Spare model*: As represented in Fig. 4 (a), we assume the considered module has m separated identical parts which can function with at least n parts. In the redundancy method, an r extra spare parts are added in the design stage. The *Self-configuration* can be modeled without extra parts. In fact, it has $n < m$ and allows the system to fail at most $(m - n)$ submodules.
- *Model 2 - Fault-reduced model*: This model is aimed for fault-reduced systems. The reducing of fault-rate can be given by a special technique (e.g., error correcting code [21]).

This model can help applying the other former analyses (physical-level or system-level) to the new system.

- *Model 3 - Error handling model*: As shown in Fig. 4 (b), this model is designed for error detection and management modules. The detection module also adds a new rate to the overall system. The fault-rate of the original module can be reduced by using *Model 1* or *Model 2*.

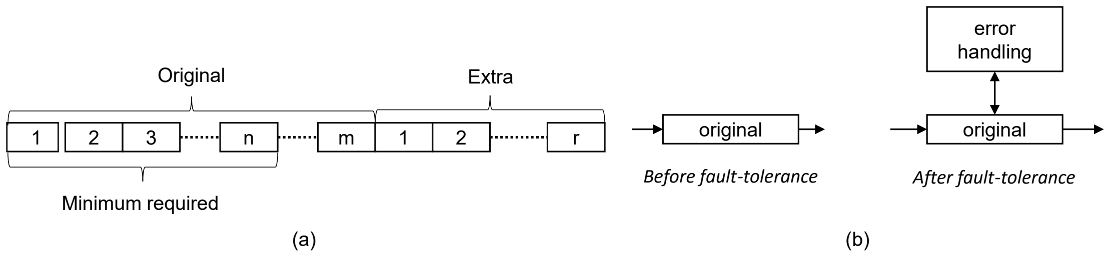


Figure 4. Classified Model: (a) Model 1 - Spare, (b) Model 3 - Error handling.

3.2. Quantitative Reliability Assessment

Dividing:

1. A Network-on-Chip consists of N_R routers.
2. A router is divided into several modules: $M_0, M_1, ..., M_a$.
3. Each module can be modeled as one of the predefined models (Model 0-3).

Conquering:

1. For each module $M_i = M_0, M_1, ..., M_a$ of a router, analyze it using one of the given strategies:
   - If the module M_i is not fault tolerant (Model 0), it is given reliability values by Eq. 3.
   - If the module M_i is a spare model or a reconfigurable module (Model 1), it is given a failure rate by *Strategy 1*.
   - If the module M_i is reduced failure by applying a special technique (Model 2), use *Strategy 2*.
   - If the module M_i is a typical fault-tolerant module (Model 3), after applying *Strategy 1 or 2* its final failure rate is given by calculation in *Strategy 3*.

Merging:

1. A router reliability is obtained by *Router Merging*.
2. A network reliability is obtained by *Network Merging*.

Figure 5. Reliability Assessments for Fault-Tolerant Network-on-Chip.

Figure 5 shows the three main steps that are necessary to obtain a comprehensive reliability assessment: A network is divided into routers which are divided into modules (*Dividing*). After dividing, the modules are analyzed and classified according to their appropriate model, and the suitable strategy is applied (*Conquering*). The final reliability is obtained by merging all modules together (*Merging*).

3.2.1. Conquering

We consider the components of a router by using the following strategies which are applied to each one of the four basic models presented in the previous section.

Strategy 0: Applied for *Model 0 - Non-fault tolerant model* where if the module is not fault-tolerant, its failure rate is simply estimated using Eq. 3.

Strategy 1: Applied for *Model 1 - Spare model.* This strategy handles hard faults using spare modules or by reconfiguring an alternative part.

We assume that the considered module has m separate identical parts and can function with at least n parts. In order to enhance the reliability, extra r spare parts are added in the design stage. f is the number of parts that are faulty in a state. The Markov state model can be built as shown in Fig. 6. Each state is labeled with the number of healthy parts and the failure-rate is indicated by Eq. 3. The original system consists of m parts, and its MTTF can be expressed as:

$$\mathbf{MTTF}_{original} = \frac{1}{m}\frac{1}{\lambda_{single-part}} \tag{4}$$

By applying Eq. 1 based on the probability of healthy states, the MTTF can be expressed as:

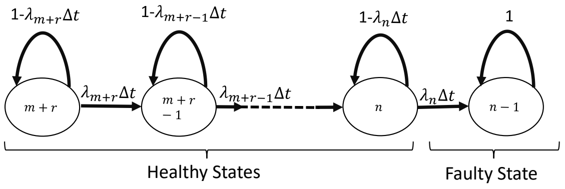


Figure 6. A Markov-state reliability model for spare modules.

$$\mathbf{MTTF}_{FT} = \Sigma_{i=n}^{m+r}\frac{1}{\lambda_i} = \frac{1}{\lambda_{single-part}}(\Sigma_{i=n}^{m-1}\frac{1}{i} + \frac{1}{m} + \Sigma_{i=m+1}^{m+r}\frac{1}{i}) \tag{5}$$

The RAF values can be calculated as follows:

$$\mathsf{RAF}_{model-1} = \frac{\mathbf{MTTF}_{FT}}{\mathbf{MTTF}_{original}} = \Sigma_{i=n}^{m+r}\frac{m}{i} = 1 + \Sigma_{i=n}^{m-1}\frac{m}{i} + \Sigma_{i=m+1}^{m+r}\frac{m}{i} \tag{6}$$

3.2.2. Strategy 2

This strategy is designed for *Model 2 - Fault-reduced model.* In this case, the efficiency can be predicted from other analyses or probability calculations. With a fault reduction value f_{FT} given by the technique, the new fault rate is obtained by Eq. 7

$$\lambda_{FT} = f_{FT}\lambda_{original} \tag{7}$$

Where f_{FT} is the inverse value of RAF:

$$f_{FT} = \frac{1}{RAF} = \frac{\lambda_{FT}}{\lambda_{original}} \tag{8}$$

3.2.3. Strategy 3

This strategy is applied to *Model 3 - Error handling model.* As previously mentioned, in prior strategies we demonstrate the efficiency of the fault-tolerance techniques using analytical models [72]. We model both original and fault-tolerant systems to have two Markov state models as represented in Fig. 7 (a) and Fig. 7 (b), respectively, where:

- S_0 is the initial state.
- S_F is the faulty state of the original system.
- S_1 is the faulty state of the original system which can be corrected by the fault-tolerant technique.
- S_2 is the faulty state of the original system which cannot be corrected by the fault-tolerant technique.
- S_{C-F} is the faulty state of the repair module.

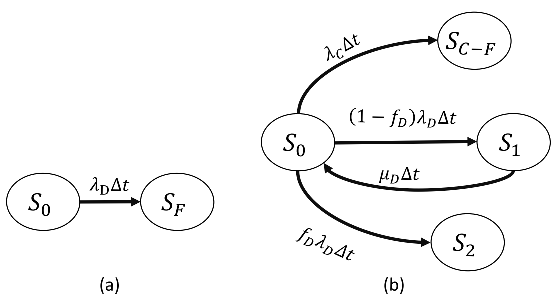


Figure 7. A simplified Markov-state reliability model for (a) the original system; (b) the fault-tolerant (FT) system.

Because of the protection from the fault-tolerant technique, the FT system can handle some faults. Therefore, we define the transition rates as:

- λ_D is the fault-rate of the original system (D).
- λ_C is the fault-rate of the repair module of the FT system.
- μ_D is the repair-rate which is provided by the repair module (C) on the original system (D).
- f_D is the fault reducing value by applying the fault-tolerance mechanism.

Based on the Markov state of the two systems, as shown in Fig. 7, the reliability function is givens as:

$$R^*(s) = P_{S0} \tag{9}$$

The final fault-rate of the fault-tolerance system is as follows:

$$\lambda_{FT} = f_D \lambda_D + \lambda_C \tag{10}$$

The RAF value can be then expressed as:

$$\mathsf{RAF}_{FT} = f_D + \frac{\lambda_C}{\lambda_D} \tag{11}$$

3.2.4. Merging

After analyzing all modules, the system reliability is calculated based on its sub-modules. Therefore, we start first by merging the router components and then merge all the network components to obtain the entire system reliability.

Router merging

We first determine the router is reliable only if it is able to transmit correctly a given data from any input to any output port. By applying equation 3, the fault rate of a router is obtained as follows:

$$\lambda^*_{router} = \sum_{i=1}^{N} f_{M_i} \lambda_{M_i} \tag{12}$$

Where λ_{M_i} is the fault-rate of module M_i and f_{M_i} is the fault reduction rate given by attaching this module to the system.

Network merging

The next step is to evaluate the network reliability. Among several existing network's reliability terminologies [52], this work uses "all-terminal reliability" for the analysis. "All-terminal reliability" is defined as all PEs are connected to the network. In other words, all PEs have the ability to communicate with any PE in the network. According to [52, 60], the most common method is using *Node-based reliability*. To analyze the network's reliability, we analyze the possible failure cases that may corrupt the system. We define the major failure cases as follows:

- Failure on local connection: a failure on the connections, which are handled by NIs, between routers and PEs can corrupt the network's reliability. This involves two channels (input and output) and an input buffer which is constantly attached to its input channel.
- Failure on transmitting path: a failure on the transmitting path can corrupt the network's reliability. The transmitting path is considered as a set of connections between routers.
- Failed on other router modules: A failure in non-transmitting parts (e.g., switch allocator, management modules) of a router may malfunction the router.

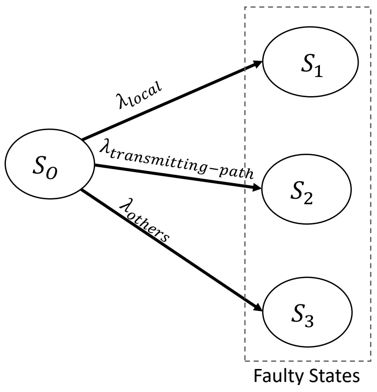


Figure 8. A Markov state of a mesh-based network.

From the failure cases, a Markov state model is built as shown in Fig. 8. As a result, the fault rate of a network of N_R routers is given as follows:

$$\lambda_{network} = \lambda_{local} + \lambda_{transmitting-path} + \lambda_{others} \tag{13}$$

Where:

- $\lambda_{local} = N_R \times (2\lambda_{1-channel} + \lambda_{input-buffer})$ is the fault-rate of all local connections. N_R is the number of routers in the network.
- $\lambda_{transmitting-path}$ is the fault-rate of a transmitting paths between routers inside the network. Designers can estimate this value based on analytical analysis or simulation model proposed in [52] [60] [50] [73]. In here, we apply *k-failure* [73] model to assess the reliability.
- λ_{others} is given by the fault-rates of other parts (non-routing parts) of routers.

In this work, we consider $\lambda_{transmitting-path} = \lambda_{RTR} \times N_R$. λ_{RTR} is the fault-rate of a router-to-router (RTR) connection which represents one node connection from a router to any adjacent router. The RTR connection failure rate depends on the position of the router in the network. Here, we use the *k-failure* [73] model: a router is disconnected at the presence of *k* failures. We adopted this model with a modification: the failure value *k* is defined as a connection and it depends on the router's position. For example, the corners, the edges, the side and the middles of the 3D Mesh NoCs have three, four, five and six routing selections, respectively. This condition is the maximum value that fault-tolerant routing algorithms can achieve. With the fault assumption in Eq. 3, the routers located at a similar position (corner, edge, side or middle) have a similar fault rate. Therefore, the failure rate of RTR connection is expressed as follows:

$$\lambda_{RTR} = P_{corner} \times \lambda_{corner} + P_{edge} \times \lambda_{edge} + P_{side} \times \lambda_{side} + P_{middle} \times \lambda_{middle} \tag{14}$$

Where the failure rates: λ_{corner}, λ_{edge}, λ_{side}, and λ_{middle} are obtained from the fault-rate of a connection. P_{corner}, P_{edge}, P_{side} and P_{middle} are the probability of having a router in corner, edge, side and middle of the network, respectively. A connection is defined as a data path from input buffer to the next input buffer or NI's buffer. A connection consists of an input-buffer, a crossbar link, an inter-router channel. Because these components are independent, the fault-rate of a connection ($\lambda_{conn.}$) is defined as follows:

$$\lambda_{conn.} = \lambda_{1-input-buffer} + \lambda_{1-crossbar-link} + \lambda_{1-router-channel} \tag{15}$$

In order to compute λ_{RTR}, the fault rate of each position in Eq. 14 can be calculated by using Eq. 5 of Strategy 1 as follows:

$$\lambda_{position} = \frac{1}{\mathbf{MTTF}_{position}} = \frac{1}{\sum_{i=n}^{m+r} \frac{1}{\lambda_{conn.}}} \tag{16}$$

Where the identical part is a connection (its fault-rate is $\lambda_{conn.}$), r=0, n=1, and m= 3, 4, 5 and 6 for the *corner*, *edge*, *side*, and *middle*, respectively. The non-fault tolerant routing fault-rate (from MTTF) can be calculated using Eq. 4.

4. TSV-FETO Architecture

In this part, we present the TSV-FETO architecture - a highly reliable 3D-NoC which can handles soft errors, hard faults, and TSV defects. Figure 9 depicts the block diagram of the TSV-FETO router which corresponds to the final architecture combining the soft error, hard fault and TSV defect resiliency. To handle the cluster-TSV defect, we divide TSVs of a router into four clusters as shown in Fig. 9(a). As a comprehensive fault-tolerant design, TSV-FETO router includes soft error and hard fault tolerant techniques which are previously proposed in [18,24,32]. Moreover, TSV defect is also dealt in Section 5.

4.1. Soft Error Resiliency Approach

The proposed system handles the soft error resiliency by using two mechanisms: (1) *Pipeline Computation Redundancy* to handle soft errors on the pipeline stages, and (2) *The Error Correcting Code* and *Automatic Retransmission Request* to deal with the soft errors on the data path.

4.1.1. Pipeline Computation Redundancy

The principle of the proposed soft error handling method relies on a solution called *Pipeline Computation Redundancy* (PCR) [17, 24, 74]. The Next Port Computation (NPC) and Switch Allocation (SA) run in parallel which is achieved by the LAFT routing algorithm (discussed in more details in Section 4.2.3) where the dependency between the two stages is eliminated. In order to handle the soft error in these stages, the computation redundancy is made for them. After the first computation, the two stages have an additional computation clock cycle. If a soft error is detected, the pipeline stages are re-executed the third time, and a majority voting is used to handle the error.

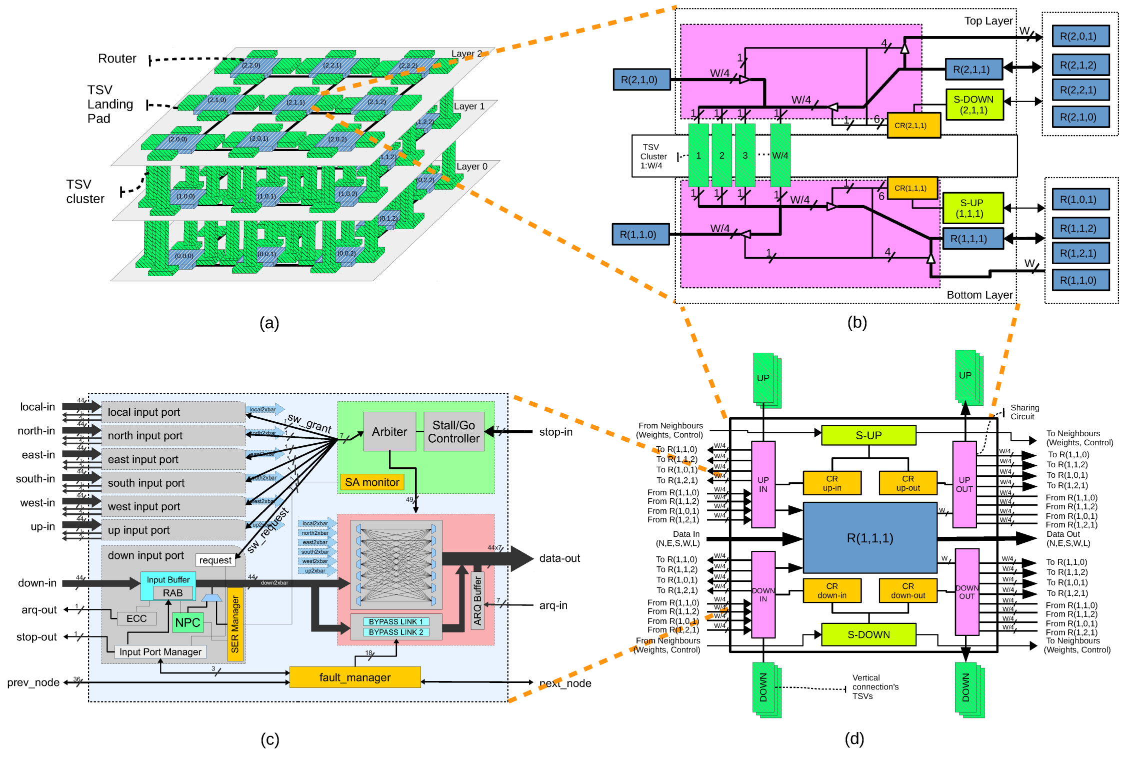


Figure 9. TSV-FETO router architecture: (a) An example of 3 × 3 × 3 NoC; (b) Cross-layer interface between two routers; (c) Router architecture; (d) Wrapped router architecture with TSV fault-tolerance support.

4.1.2. Error Correction Code

In this work, we use an ECC, which is Single Error Correct Double Error Detection (SECDED) [75], to deal with soft errors, and to support the detection and diagnosis mechanisms, as explained later. Moreover, we adopted the HARQ [20] mechanism, which allows retransmitting the error flit as a flow-control. If the faulty bit cannot be corrected but can be detected by the SECDED code, an ARQ signal is sent to the sender to request a retransmission.

4.2. Hard Fault Tolerant Approach

4.2.1. Random Access Buffer

The *Random Access Buffer* mechanism (RAB) [18] can recover faults in the input-buffer. When a fault is detected in one of the slots, the *input port manager* ignore the defected slots when assigning the write and read addresses.

This is a preprint of a book chapter in Horizons in Computer Science Research, Volume 16, Chapter 7, pp 199-246. URL: `https://www.novapublishers.com/catalog/product_info.php?products_id=64154`. ISBN: 978-1-53613-327-1. Copyright (c) 2018 Nova Science Publishers.

4.2.2. Bypass Link on Demand

The *Bypass Link on Demand* mechanism (BLoD) [18] provides additional escape channels whenever the number of faults in the baseline 7x7 crossbar increases. In the case where a fault is detected in one or several crossbar links, the *fault_manager* disables the faulty crossbar links and enables the appropriate number of bypass channels.

4.2.3. Fault Tolerant Routing Algorithm

In this work, the Look-Ahead-Fault-Tolerant [32] (LAFT) routing algorithm is adopted. As a look-ahead routing [76], the current node calculates the routing path for the next node. It first receives the faulty link information from the following node and decides the suitable routing path. The algorithm also favors the routing paths to the middle of the network which gives higher diversities.

4.3. Light-Weight DDRM Mechanism

The proposed *Detection, Diagnosis, and Recovery Mechanism* (DDRM) [17] uses the feedback from the ECC (Error Correction Code) and the HARQ (Hybrid Automatic Retransmission Request) protocol to monitor the errors. The condition of hard fault detection is the occurrence of consecutive ARQs. Since soft errors are temporary which can be recovered by re-transmission. Therefore, if re-transmission cannot correct the fault, it is considered as a hard fault.

For the diagnosis and recovery phase, the router's *Fault-manager* module initiates the diagnosis with input buffer checking. If there is one buffer slot repeatedly has multiple ARQs, this slot is faulty. If all slots have multiple ARQs, the following path (crossbar and channel) is defective.

If the *fault-manager* indicates that the fault may belong to the crossbar or the inter-router channel, it firstly configures the *Bypass-Link-on-Demand* (explained in the previous subsection) to establish an alternative crossbar link. Then, if the flit is found to be not faulty by the ECC module, the *fault-manager* concludes that the fault is in the crossbar which is already handled by the BLoD mechanism. Otherwise, the inter-router channel is faulty, and the LAFT routing is used to avoid the defective/faulty link.

5. TSV-Cluster Defect Tolerance

In the last two sections, we briefly present how the proposed system can work around both soft errors and hard faults. Notice that our system has built-in ECC module which can easily detect and correct the random TSV defects. Therefore, this section presents the technique to handle the TSV-cluster defects in 3D-NoCs. Our solution is to share TSVs between neighboring routers. Thus, when a TSV-cluster fails, its router can borrow a healthy cluster from one of its neighbors to maintain the connection. Moreover, we also present several design optimizations to improve the reliability of the system (Section 5.5).

5.1. Fault Assumptions

Before we present the system structure, this subsection clarifies the fault assumptions taken in this proposal. Because of the cluster defect [11, 37, 39] is the major obstacle to be dealt with in this work, we assume there are no random defects. Here, we consider an occurred fault makes the whole TSVs in the cluster defected. For the random TSV defect, our built-in ECC modules can handle a limited number of them. For a high number of random defects in a cluster, using redundancy [13, 38, 44, 77] can be easily integrated into our TSV-cluster design. For controlling signals using TSVs, they are considered as a part of the TSV cluster instead of separated TSVs, which are better dealt as a random defect (e.g., [43] uses Double TSV [77]). The detection process, which may need a Built-In-Self-Test module [78,79], is assumed to be done by DDRM, where the vertical link could detected as faulty or not.

5.2. System Structure

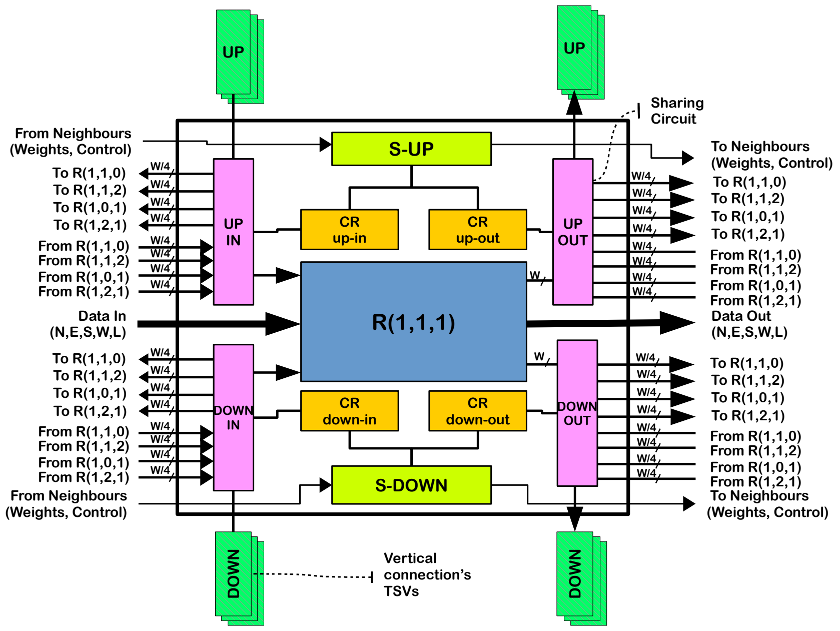
A simplified layout example of $3 \times 3 \times 3$ 3D-NoC system using the proposed TSV usage is depicted in Fig. 9(a). For each vertical connection, a router needs a set of TSVs. Here, they are divided into four groups. As a result, a router owns four TSV-clusters and has a maximum of four nearby TSV-clusters. If a TSV-cluster of a router defects, the router can choose one of its four neighboring clusters as a replacement without the need for redundancy. To satisfy the timing constraints, the router chooses the closest TSV-cluster among its neighbor clusters. Taking into account further TSV-clusters is not considered to avoid long wires that are needed to establish the connection. By structuring the TSVs into four clusters for each router, we can maintain the scalability of 3D-NoCs and avoid long wire delay.

5.3. Sharing Circuit Design

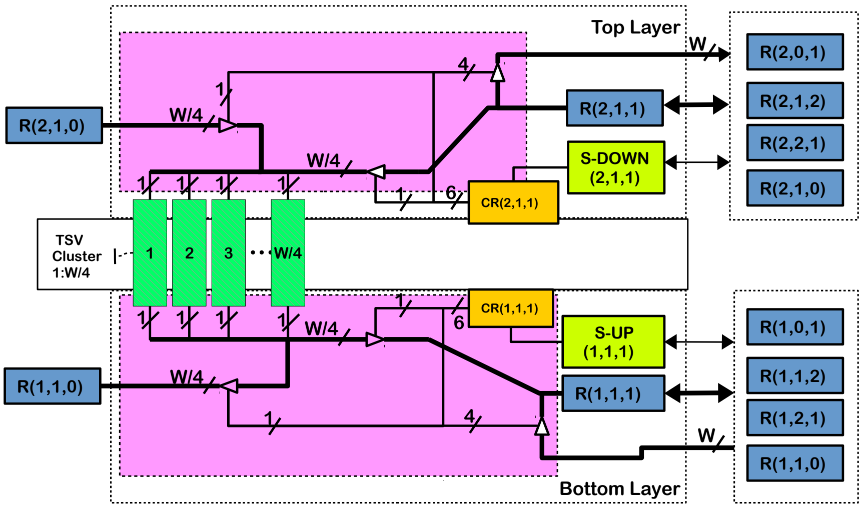
To borrow a TSV-cluster from a neighbor, the router needs a supporting module. Figure 10 (a) shows the wrapper of a 3D-Router with the additional supporting modules that perform the sharing algorithm, later explained in Section 5.4. There are two identical sharing modules (S-UP and S-DOWN) for the two vertical up and down connections and each connection has two configuration registers (CR) for the input and output ports. As previously depicted in Fig. 10(b), *R(1,1,1)* shares the TSV-clusters with its four neighbors: *R(1,1,0)*, *R(1,1,2)*, *R(1,0,1)*, and *R(1,2,1)*. Figure 10 (b) shows the sharing circuit for a TSV-cluster. The input of this TSV-cluster is shared between *R(2,1,0)* and *R(2,1,1)* on layer2. The output of this TSV-cluster is shared between *R(1,1,1)* and *R(1,1,0)* on layer1. In the case where this TSV-cluster is defected, or borrowed, the data can be sent by using one of the four neighboring clusters.

5.4. Adaptive Online Sharing Algorithm

In the previous section, we presented how a router can use its nearby TSV-clusters to maintain the connection and the operation of a layer. The CR values need to be configured to deal with the TSV defects. The simplest way for this process is to perform it offline, and the configuration fuses the TSV group [39]. However, fixing the connections has two main drawbacks: (1) recovering a newly defected TSV needs to halt the system and re-perform the mapping, and (2) each



(a)



(b)

Figure 10. The TSV fault-tolerance architecture: (a) Router wrapper; (b) Connection between two layers. Red rectangles represent TSVs. *S-UP* and *S-DOWN* are the sharing arbitrators which manage the proposed mechanism. *CR* stands for configuration register and *W* is the flit width.

application has a different distribution in the vertical connections and variations depending on the running task which is not optimized by offline mappings. Consequently, we aim to perform the mapping online so that the system can react immediately to the newly defected TSV-clusters and can consider the connectivity of the 3D-NoC system. Thus, this subsection provides an

ALGORITHM 1: TSV Sharing Algorithm.

```
// Weight values of the current router and its N neighbors
Input: $Weight_{current}, Weight_{neighbor}[1:N]$
// Status of current and neighboring TSV-clusters
Input: $TSV\_Status_{current}[1:N], TSV\_Status_{neighbor}[1:N]$
// Request to link TSV-clusters to neighbors
Output: $RQ\_link[1:N]$
// Current router status
Output: $Router\_Status$
foreach $TSV\_Status_{current}[i]$ do
    if $TSV\_Status_{current}[i]$ == "NORMAL" then
        // It is a healthy TSV-cluster
        $RQ\_link[i]$ = "NULL"
    else
        // It is a faulty or borrowed TSV-cluster
        find c in 1:N with:
        $Weight_{neighbor}[c] < Weight_{current}$
        $Weight_{neighbor}[c]$ is minimal
        and $TSV\_Status_{neighbor}[c]$ == "NORMAL";
        if (c==NULL) then
            return $RQ\_link[i]$ = "NULL"
            return $Router\_Status$ = "DISABLE"
        else
            return $RQ\_link[i]$ = c
            return $Router\_Status$ = "NORMAL"
        end
    end
end
```

online algorithm for sharing TSVs which can be implemented into the system.

The Algorithm 1 shows the proposed algorithm for our sharing mechanism. Each router is assigned to a weight for each of the vertical connections. This weight decides its priority in sharing/borrowing. The weight can be assigned at the design process or can be updated by a dedicated module. Changing the weights of routers can create different mappings. At the initial stage, all routers in the network exchange their weights and their TSV-clusters status with their neighbors. In the next step, the algorithm performs the mapping process. If TSV-cluster defects, its corresponding router should find from its neighbors a possible candidate by relying on the following conditions:

- The weight of the candidate has to be smaller than the current router.
- The candidate TSV-cluster has to be healthy and not borrowed.
- The weight of the final candidate is the smallest among all the possible candidates.

At the end of the algorithm, the router finds out the possible candidate for borrowing. If no candidates were found, the router's vertical connection is disabled. If there is a candidate, the router sends a request to the borrowing router to use its TSV-cluster as a replacement for the defected one. The routers having borrowed TSV-clusters also look for a replacement among one of their neighbors. By using a weighted system, the disabled TSV-clusters focus on smaller weight routers. As a result of the algorithm, chains of sharing are created that disable the routers on the edges. Instead of having ten defected TSV-clusters, the algorithm only disables six routers having the lowest weights (40% of reduction). Consequently, maintaining the connections of the center routers, which have higher weights and utilize more vertical communications, can reduce the impact of TSV defects regarding the overall performance.

5.4.1. Weight Generation

One of the most important parameters in the sharing algorithm is the weight values of the routers. The weights help the algorithm decide what router is suitable to be borrowed. As a result of Algorithm 1, the routers having smaller weights are disabled after the chains of sharing are established.

Because the weights decide the priority of the routers in the sharing process, they need to be optimized to obtain a maximum system performance. In order to do that, the best solution is using a statistic based solution where the priority of the vertical connection depends on the communication traffic [80, 81]. In other words, the vertical connections having more data transmissions are assigned higher weights; otherwise, smaller weights are assigned. Because application mapping is out of the scope of this work, we adopt a simple method where the routers in the middle of the layer have the highest weights. The router's weights are decreased and become the lowest at the edges of the layer. Equation 17 shows the used weight value assignment. The output of this weight assignment on a layer of 4×4, for instance, the weights of routers *R(1,0,0), R(1,1,0)*, and *R(1,1,1)* are 1, 2, and 3, respectively.

$$\text{Weight}_{\text{router}}(x, y) = \min(x, \text{cols} - x) + \min(y, \text{rows} - y) + 1 \tag{17}$$

5.4.2. Weight Adjustment

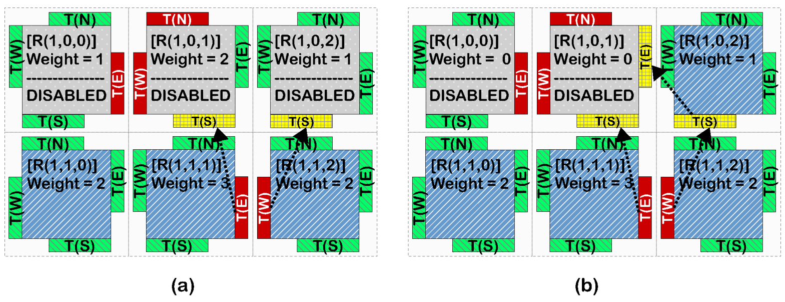


Figure 11. Example of the weight adjustment performed to disable routers' sharing: (a) Before weight update; (b) After weight update.

After applying the sharing mechanism, the disabled TSV-clusters are shifted to the region which consists of low weighted routers. Figure 11 (a) shows a case of three routers (*R(1,0,0), R(1,0,1)* and *R(1,0,2)*) which are disabled after the sharing process. However, there are still a chance of optimizing these routers to obtain a better mapping. In fact, *R(1,0,2)* can borrow a TSV-cluster from *R(1,0,1)*. Therefore, the number of TSV-clusters of *R(1,0,2)* can be maintained to four. Algorithm 2 shows the weight adjustment algorithm. It first calculates the total number of healthy TSVs that are possible for using. If the total number of healthy TSV-clusters is larger

ALGORITHM 2: Weight Adjustment Algorithm.

```
// Status of current and neighboring TSV-clusters
Input: $TSV\_Status_{current}[1:N], TSV\_Status_{neighbor}[1:N]$
// Current and neighboring routers status
Input: $Curr\_Status, Neighbor\_Status[1:N]$
// Request to link TSV-clusters to neighbors
Output: $Weight_{current}$
$Curr_{TSVs} = 0$;
foreach $TSV\_Status_{current}[i]$ do
    if $TSV\_Status_{current}[i] == “NORMAL''$ then
        $Curr_{TSVs} ++$;
    end
end
$Neighbor_{TSVs} = 0$;
foreach $TSV\_Status_{neighbor}[i]$ do
    if $TSV\_Status_{neighbor}[i] == “NORMAL''$ and $Neighbor\_Status[i] == “DISABLED''$ then
        $Neighbor_{TSVs} ++$;
    end
end
// If there is at least 4 cluster, run the sharing algorithm
if $Neighbor_{TSVs} + Curr_{TSVs} >= 4$ then
    call TSV_Sharing()
end
else
    // Reduce the current weight to allow the neighbors borrow
    $Weight_{current} = 0$;
end
```

or equal than four, which is enough to maintain the vertical connection, the neighboring routers' weights are reduced. After that, the TSV sharing algorithm (Algorithm 1) is performed, where the router now can take TSV-clusters from the routers having higher weights, but is disabled.

5.5. Design Optimization

Without adding redundancy, borrowing TSV-clusters to work around the defected ones makes some routers to have less than four accessible clusters. As a result, the communication of these routers has been disabled. To tackle this problem, the naive solution is using a fault-tolerant routing algorithm to re-route the packets to a neighboring router. As we mentioned in Section 2, this solution may lead to non-minimal routing and congestion in the network. Therefore, we propose *Virtual TSV* to help these routers maintaining the connection without using any fault-tolerant routing algorithm. In the case where the *Virtual TSV* is unable to be performed, we also implement the *Serialization* technique which helps the vertical connection establishing only one or two TSV-clusters.

5.5.1. Virtual TSV

When a router is not granted the access to four TSV-clusters, it is disabled. However, if the number of nearby TSVs is larger or equal than four, which is enough for maintaining vertical communication, they can be utilized to establish a connection. A possible connection, which requires four TSV-clusters, may need clusters belonging to the neighboring routers. If these

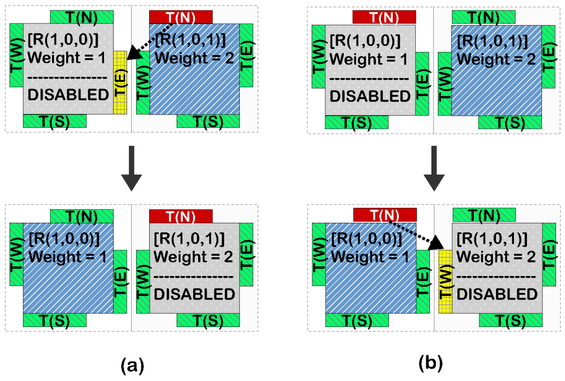


Figure 12. Examples of Virtual TSV: (a) return the TSV-cluster to the original router; (b) borrow a cluster from a higher weight router.

routers do not use these clusters, the disabled router can borrow them for a short period to establish communication.

Figure 12 (a) shows an example of how *Virtual TSV* works where *R(1,0,1)* has a defective cluster (T(N)) and borrows a cluster from *R(1,0,0)*. Because *R(1,0,0)* is unable to find any replacement for the borrowed cluster (T(E)), it is disabled. When *R(1,0,0)* needs to establish an inter-layer communication, it needs to find at least four TSV-clusters. Assuming that *R(1,0,1)* does not use the borrowed cluster T(E), it is temporarily returned to *R(1,0,0)*. When the packet is completely transmitted, the borrowing cluster is taken back by the router *R(1,0,1)* again.

On the other hand, Fig. 12 (b) shows the case where a disabled router *R(1,0,0)* temporarily borrows a TSV-cluster from a higher weight router *R(1,0,1)* to establish an inter-layer connection. For selecting a suitable candidate to temporarily borrow, Algorithm 1 is utilized.

Because there is a case where *R(1,0,1)*, which has the higher priority, occupies the TSV for a long transmission time, *R(1,0,0)* is unable to access the TSV to establish a connection. Moreover, at a high defect-rates, *R(1,0,0)* may not find any suitable candidate for virtual TSV. In order to solve these issues, we adopt the *Serialization* [82] technique to maintain the connection.

5.5.2. Serialization Technique

Although the *Virtual TSV* can help the disabled router maintaining its vertical connection, there are still two situations where *Virtual TSV* cannot be performed: (a) there are less than four healthy TSV-clusters, (b) the candidate TSV-cluster is occupied constantly by a higher priority router. In order to solve these cases, we use the *Serialization* technique [82] to maintain the connectivity.

6. Evaluation Results

6.1. TSV-FETO Evaluation

6.1.1. Evaluation Methodology

As previously mentioned, TSV-FETO system incorporates the proposed soft error resilient techniques, hard fault schemes, and TSV defect tolerance to form a highly reliable 3D-NoC system. The TSV-FETO was designed in Verilog-HDL, synthesized and prototyped with commercial CAD tools and VLSI technology, respectively. We evaluate the hardware complexity of the TSV-FETO router in terms of area utilization, power consumption (static and dynamic) and speed. To evaluate the performance of the proposed system, we select both synthetic and realistic traffic patterns as benchmarks. For synthetic benchmarks, we select Transpose [83], Uniform [84], Matrix-multiplication [85, 86], and Hotspot 10% [87]. For realistic benchmarks, we choose H.264 video encoding system [88], Video Object Plane Decoder (VOPD), Picture In Picture (PIP) and Multiple Window Display (MWD) [89]. The simulation configurations are depicted in Table 3.

Table 3. Simulation configurations.

Benchmark	Matrix	Transpose	Uniform	Hotspot	H.264	VOPD	MWD	PIP
#Tasks	-				11	4	8	4
Network Size (x,y,x)	$(6,6,3)$	$(4,4,4)$	$(4,4,4)$	$(4,4,4)$	$(3,3,3)$	$(3,2,2)$	$(2,2,3)$	$(2,2,2)$
#Packets	1,080	640	8,192	8,192	8,400	3,494	1,120	512
Packet's Size	10	10	10[a]	10	10	10	10	10

[a]For the hot spot nodes, there are additional 10% of flits.

6.1.2. Performance Evaluation

In this experiment, we evaluate the performance of the proposed architecture in terms of Average packet Latency (APL) and throughput over various benchmark programs and defect-rates. The simulation results are shown in Fig. 13 and 14. From this graph, we notice that with a 0% of defect-rate, the TSV defect and hard fault tolerant system have similar performances in comparison to the baseline system. On the other hand, soft error tolerance cause degradation in both APL and throughput. This is caused by the redundancies of the PCR technique where the routing/arbitrating processes are executed two times.

When we increase the defect-rates in the proposed system, it has demonstrated additional impacts on APL. At a 10% fault-rate using Matrix, Uniform, Transpose, and Hotspot 10% benchmarks, the cluster-TSV defect tolerant system increases the APL by 170.29%, 133.60%, 185.12% and 116.84%, respectively. These high impacts are due to the occurrence of bottlenecks inside the network. Because all vertical connections are utilized, Virtual TSV has caused congestion by sharing the TSV between two routers. The serialization is already a bottleneck technique. In comparison to soft error and hard fault tolerance, the TSV fault-tolerance suffer higher impacts from the bottleneck phenomenon.

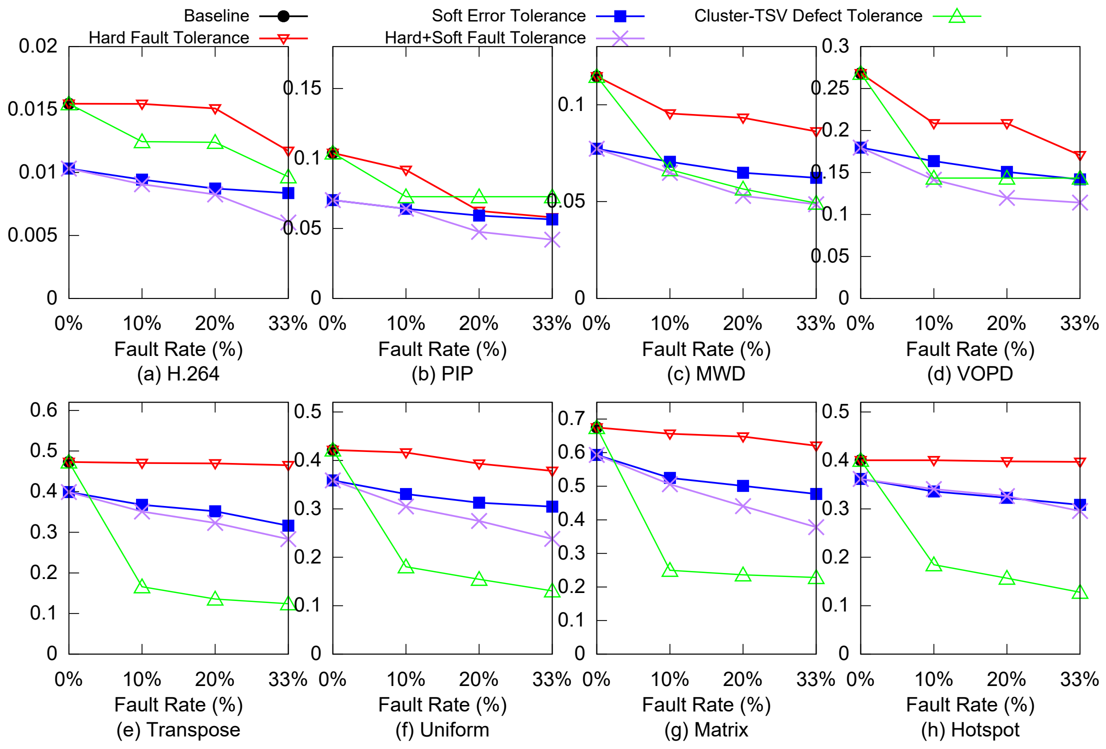


Figure 13. Average Packet Latency.

With H.264, PIP, MWD and VOPD benchmarks, the APL incrementation of the TSV fault-tolerance are significantly reduced due to the low utilization of TSV in real applications. On the other hand, a similar behavior as synthetic benchmarks can be seen in APL of a soft error and hard fault tolerant systems. We can observe the realistic benchmarks have strong impact from a 0% to a 10% defect-rates. However, when we increase the defect rate, the APL values are stable in most cases. We can observe unchanged performances by the PIP and the VOPD. The APL values of TSV fault-tolerance are also smaller than the Soft+Hard fault tolerant system while they are significantly higher in the synthetic benchmarks. This is aided by the smart and efficient borrowing mechanism and optimization of the proposed technique.

Figure 14 depicts the throughput evaluation with different benchmarks. At 0% defect-rate, the TSV fault-tolerant system's throughput is similar to that of the baseline. When defects are injected into the system, we can observe some degradation in throughput caused by the bottleneck effects on the system. Similar to APL, the throughput degradation on realistic traffic benchmarks (VOPD, H.264, MWD and PIP) is significantly better than the synthetic ones. The system at a 20% defect-rate provides a decreased throughput by 71.17%, 64.36%, 67.44% and 64.37% for Transpose, Uniform, Matrix, and Hotspot 10%, respectively. At the same defect-rate, VOPD, MWD, PIP, and H.264 have 46.03%, 50.04% 28.17%, and 19.79% of throughput degradation. This lower impact is caused by the low utilization of vertical connection rate and the optimization process. The throughput of realistic benchmarks is naturally smaller than the

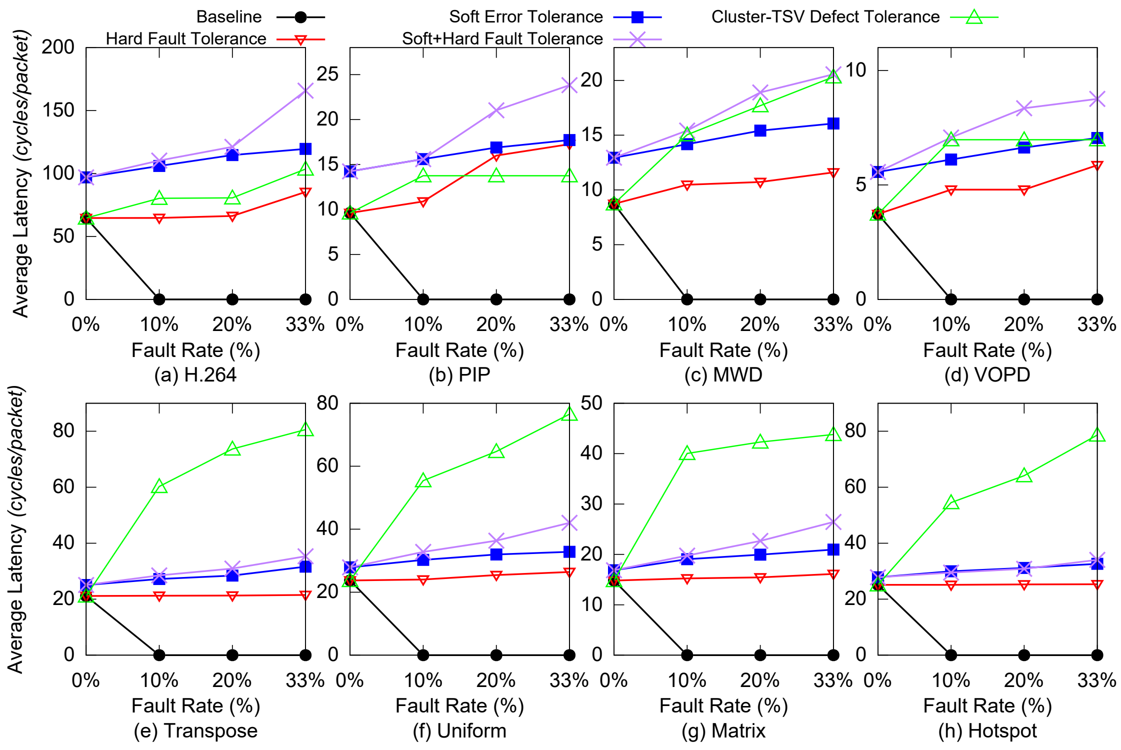


Figure 14. Throughput.

synthetic ones because of the specific tasks order of execution that was observed in the task graphs [88, 89].

Although there is a considerable degradation in the throughput evaluation, the system still maintains over 0.1 *flit/node/cycle* in the highly stressed benchmarks, even at extremely high defect-rates.

6.1.3. TSV Defect Rate Evaluation

In this section, we provide the impact of the different defect-rates. To demonstrate the scalability of the proposed architecture, we set up several layer sizes: 2×2, 4×4, 8×8, 16×16, 32×32, and 64×64. TSVs are grouped in clusters as presented in Section 5. We also vary the TSV-cluster defect-rates: from 5% to 50%. Because our technique focuses on the cluster defect, random defects are assumed to be dealt with typical redundancy methods. The position of cluster defects are generated randomly and we perform the proposed algorithms with 100,000 different samples and calculate the average results. We measure the ratio of four types routers in the layer: *Normal* (healthy or corrected), *Virtual* (router with virtual TSV), *Serial* (router using serialization) and *Disabled* (disabled routers). We also compare the obtained results with *"Normal w/o FT"* (Normal without Fault-Tolerance), where no fault-tolerance method is used and the vertical router connection having defects is disabled.

As shown in Figure 15, the system mostly operates without disabling any vertical connec-

This is a preprint of a book chapter in Horizons in Computer Science Research, Volume 16, Chapter 7, pp 199-246. URL: `https://www.novapublishers.com/catalog/product_info.php?products_id=64154`. ISBN: 978-1-53613-327-1. Copyright (c) 2018 Nova Science Publishers.

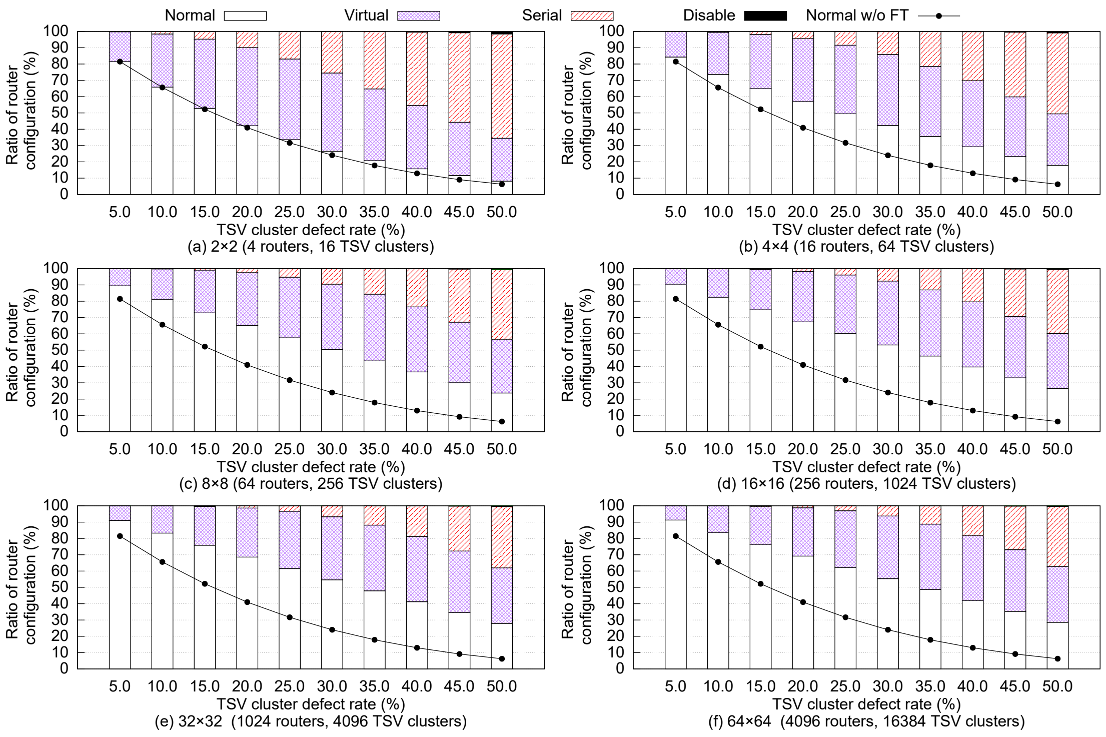


Figure 15. Defect-rate evaluation.

tions with fault-rates under 50%. Thanks to the *Virtual TSV* and *Serialization* techniques, the routers having less than four clusters are still able to work. Even at less than 20% of defect-rate, there are less than 10% of serialization connections in all simulated layer sizes. With 50% of defect-rate and a 2×2 layer size, the disabled router rate is negligible with about 1.565%. This can be easily dealt using a light-weight fault-tolerant routing algorithm. When the layer size increases to be larger than 8×8, the number of disabled connections is mostly insubstantial. At 50% defect-rate, the disabled router ratio is nearly 0.63%, 0.50%, 0.44% and 0.42% with 8×8, 16×16, 32×32, and 64×64 layer sizes, respectively. However, these defect-rates are extremely high; thus, our proposed mechanism can be considered as a highly reliable.

In comparison to the system without fault-tolerant methods, there is a significant improvement in terms of healthy connections, especially at large layer sizes. In Figure 15, the percentage of routers having four healthy TSV-clusters is represented by the "Normal w/o FT" curve. At 50% defect-rate, the average ratio of normal routers has been improved by 29.83%, 186.26%, 280.76%, 324.42%, 346.74%, and 257.79% for 2×2, 4×4, 8×8, 16×16, 32×32, and 64×64 layer sizes, respectively. The improvements are lesser with small layer sizes such as: 2×2 or 4×4. However, thanks to the Virtual TSV and Serialization, the workable connection rates have nearly reached 100%.

In summary, this evaluation has shown a significant improvement in terms of reliability provided by our proposed mechanism. Thanks to the efficiency of the proposed architecture and algorithm, the system can mostly maintain all vertical connections, even at extremely high defect-rate (50%). This evaluation also shows the proposed mechanism ability to remain efficiently scalable. The proposal can be applied from a small layer size (e.g., 2×2) to a larger one (e.g., 64×64). The evaluation is also performed with a solid number of tests (100,000) which strongly demonstrates the efficiency of the proposed approach. There were some cases where some routers were disabled; however, they can be recovered by simple and light-weight fault-tolerant routing algorithms.

6.1.4. Hardware Complexity Evaluation

In our final evaluation, we considered the hardware complexity of the proposed TSV-FETO. We start first by observing the additional hardware added to the baseline system when we employ the hard fault tolerance model. Then, we evaluate the impact when we consider the soft error tolerant model. Finally, we assess the entire TSV-FETO system including both soft and hard fault mechanisms. The configurations of the network are shown in Table 3. Table 4 illustrates

Table 4. Hardware complexity comparison results between the baseline and the proposed TSV-FETO router.

Model	Area (μm^2)	Power (mW)			Speed (Mhz)
		Static	Dynamic	Total	
Baseline router [90]	18,873	5.1229	0.9429	6.0658	925.28
3D-FTO router [18]	20,104	6.6037	1.2331	7.8368	909.09
3D-SET router [17]	28,418	9.9071	2.5710	12.4781	625.00
3D-FETO router [17]	30,477	10.2576	2.6910	12.9486	613.50
TSV-FETO	38,838	11.7910	2.8273	14.6128	537.63

the hardware complexity results of TSV-FETO in terms of area, power (static, dynamic, and total), and speed. In the hard fault tolerant router (3D-FTO), the area and power consumption overheads have increased by 6.52% and 29.19%, respectively. The maximum speed has also slightly decreased. On the other hand, our soft error handling mechanism adds seven ARQ buffers and some combinational logic which increase the area cost (50.57%) and power consumption (105.71%) more significantly. However, the proposed TSV-FETO model introduces 7.25% and 3.77% extra area and power consumption, respectively, when compared to the soft error tolerant router. In comparison to the baseline model, TSV-FETO increases the area and power consumption by 61.48% and 134.69%, respectively, while the maximum speed decreases by 33.70%. In summary, our proposed models are penalized in terms of the area, power consumption, and maximum frequency due to additional logic and registers that are necessary for fault handling mechanisms; they provide a full resiliency against a significant amount of soft and hard faults.

6.1.5. Comparison

To understand the efficiency of the proposed approach, we compare it with existing solutions as shown in Table 5. Here, we analyze our proposal with a network size of $4 \times 4 \times 4$. Because the router and its TSV clusters structure are identical, similar results can be obtained with the others network sizes. *TSV Grouping* [37] optimized the configuration of redundancy to deal with TSV-cluster defects. *TSV Network* [39] established TSVs into networks which allow routing from defected TSVs to redundant ones. We select the best results on these two works [37, 39] for the comparison. From this table, we can see that the average area of our proposal is $151.47\mu m^2$ per TSV and, for a TSV size of $10\mu m \times 10\mu m$, the area overhead is about 51.47%. The *TSV Network* [39] has similar value for 4:2 configuration (4 original TSVs and 2 redundant TSVs).

On the other hand, the other configurations obtained lower area overheads. Nevertheless, we have to note that our arbiter not only consists of the rerouting circuit (similar to the multiplexers in *TSV Network* and *TSV Grouping*); but, also includes an online adaptive algorithm designed in hardware, in addition to the *Virtual TSV* and *Serialization* techniques. Both *TSV Grouping* and *TSV Network* have to require additional dedicated circuitry to recover from the cluster defects.

TSV Grouping demonstrated a 100% of yield rate under a defect-rate of 1% and *TSV Network* obtained nearly 100% in the most cases. However, their approaches are different than our scheme, where they add redundancy to correct the defect TSVs. As a result, if the number of defected TSVs is larger than the number of redundant ones, they are unable to recover from the defected clusters. On the other hand, our technique can significantly improve the reliability by providing 98.11% of workable routers at 50% of defected TSV-clusters. Moreover, at the low defect rates (e.g. under 5%), our proposal also ensures 100% of working connection and demonstrates small performance degradations in the realistic traffic pattern benchmarks. Even with disabled vertical connections, the reliability of our system can also be improved (i.e., covering the remaining 1.89%) by using a lightweight fault-tolerant routing which would have a negligible impact on the area overhead.

Table 5. Comparison results between the proposed approach and the existing works.

Model	**TSV Network** [39]				**TSV Grouping** [37]		**This work**
Technology	65 nm				N/A		45 nm
#TSV	1000				6000		8448
Configuration	4:2	8:2	$4 \times 4 : 8$	$16 \times 16 : 32$	4:4	20:5	$11 \times 4 \times$ 4:0
#Spare TSV	512	512	256	128	6000	1500	0
45nm **Area** (μm^2)	372 ^{2}	744 ^{2}	1,116 ^{22}	1,116 ^{2}	11,160 ^{1}	12,555^{1}	434,784^{3}
TSV's Area (μm^2)	151.572	126.244	152.316	128.03	113.916	127.09	151.47
Reliability	100%	99%	100%	100%	100%		98.11%
Fault Assumption	$(\delta_{TSV} = 0.01\%, \alpha = 2)^4$				$(\delta_{TSV} = 1\%,\alpha = 2)^4$		$(\delta_{cluster} = 50\%)^4$

[1] The authors use 2:1 multiplexers [37]. For comparison, we use the area cost of multiplexer from Nangate $45nm$ (MUX2_X1: $0.186\mu m^2$) .
[2] The authors use 1-to-3 multiplexers [39] which consists of two MUX2_X1 multiplexers ($2 \times 0.186\mu m^2$).
[3] For fair comparisons, our arbiter only consists of the TSV sharing and serialization modules as shown in Table 4.
[4] δ: defect-rate. α: parameter of Poisson distribution [37, 39].

6.2. Reliability Assessment Evaluation

In this section, we evaluate the reliability of NoC systems with two methodologies: the proposed analytical model and a system-level simulation. To demonstrate the efficiency of the proposed analytical analysis, we used our proposed NoC system as the case of study. We use the proposed TSV-FETO and its baseline model for these evaluations. To compare between the two methods, we use a similar error-rate then calculate the MTTF and RAF values.

6.2.1. Accuracy Evaluation

In order to illustrate the accuracy of the proposed assessment method, we compare it with a Monte-Carlo simulation [50]. We use the fault injection method on netlist files [51] for more accurate results. Based on the same fault assumptions as the assessment, we also calculate the MTTF and RAF values for the ease of comparison.

In order to verify the analytical model, we use weight-based and gate-based configurations. The fault-rate ratio can be seen in Table 6 where the errors are focused on input buffers, crossbar, and links.

Table 6. Router's Weight and Gate Ratio.

Module	Submodule	Weight
Network	Network	100%
	Routers	70%
	Channels	30%
Router	Router	100%
	Input Buffer	69.72%
	Crossbar	8.00%
	Switch-Allocator	7.00%
	Others	15.28%

Figure 16 shows the comparison of RAF values between the proposed analytical model and the simulation results. For a single router assessment, the analytical method predicts the RAF values with acceptable deviations (under 33%). Moreover, there is a significant amount of hidden faults in the baseline model. In this kind of situations, faults are injected; but, they do not give a detectable impact on the system. Especially, soft errors, which are injected in a single clock cycle, are likely to become hidden faults. In contrast, hard faults on the router give a higher impact on the system.

For networks' assessments, we also simulated and compared the RAF values for four different network sizes: $2 \times 2 \times 2$, $3 \times 3 \times 3$ and $4 \times 4 \times 4$. The worst cases can be observed in the soft-error simulation with weight distribution and hard-fault simulation with gate ratio distribution. However, the accuracy is still better than a single router where most of the deviations are less than 23%. The worst case is hard fault tolerance with the gate ratio distribution of a $3 \times 3 \times 3$ network where the difference can reach up to 31.64%.

In summary, the overall accuracy is acceptable with most cases having less than 33% of deviations. There are some cases where the difference is considerable; however, the deviation is logical due to the low granularity of the reliability assessment. In fact, when designers apply the

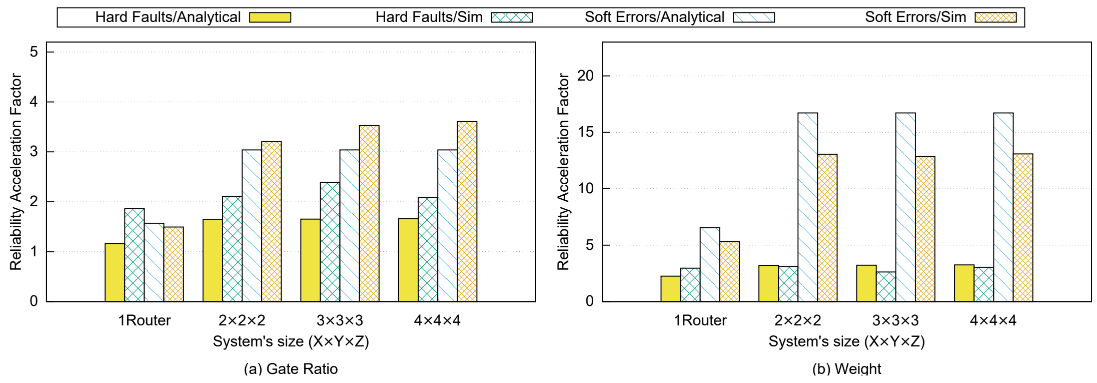


Figure 16. Comparison results.

assessment method before obtaining the design's characteristics (e.g., gate ratios), the deviations are still reasonable.

6.2.2. Reliability Assessment Speedup

The proposed analytical method offers faster estimation time in comparison to other conventional methods. Table 7 shows the reliability assessment simulation time and the speedup obtained with the proposed analytical model when compared to the conventional MTTF simulation [17]. The proposed method's calculation is performed using GNU Octave, and the MTTF simulation performs netlist simulations for 1000 cases using Cadence NCSim CAD tool. Both simulations were conducted on a Linux CentOS 6.4 machine using Intel Xeon E5-2620 (8 cores, 2.10Ghz) and 64 GB of RAM.

Table 7. Reliability Assessment Speedup.

Evaluated module	A MTTF simulation	Proposed method	Speedup
A router	11 hours	0.090 second	440,000
A $2 \times 2 \times 2$ network	20 hours	0.091 second	791,209
A $3 \times 3 \times 3$ network	2 days	0.092 second	1,878,261
A $4 \times 4 \times 4$ network	3.5 days	0.109 second	2,774,312

Thanks to the low complexity of the proposed method, the simulation time is always in the hundreds of milliseconds range for all cases. On the other hand, the MTTF simulation requires more than 11 hours of computation for just a single router. This results in a speedup of 440,000 times with our proposed method. The long execution time of the MTTF simulation is caused by the main following factors: (1) the high complexity of the netlist files where a router's netlist file consists of over 15,000 separated gates; (2) the high complexity of the fault injection (i.e., each gate requires an error injection module and a fault distribution system); (3) the verification complexity which usually tries to cover all possible operational situations (e.g., a seven-ports router has 49 (7^2) cases of communication); and (4) it requires four different simulations for four different types of fault: soft error, hard fault, stuck-at-0 and stuck-at-1. When we increase the network size, the MTTF simulation time has significantly increased. On the other hand, the assessment time of the proposed method does not scale up with the network size. This is given by the assumption that the routers in the same position (corner, edge, side or middle) inside the network have a similar fault-rate. Therefore, the calculation can be reduced into Eq. 14 and 13. In fact, the obtained speedup with the proposed method is 791209, 1878261, and 2774312 for $2 \times 2 \times 2$, $3 \times 3 \times 3$, and $4 \times 4 \times 4$ network sizes, respectively. The speedup values are expected to be much higher with larger network sizes.

In summary, the proposed analytical method provides a speedy solution to estimate the reliability of NoC systems. Although the proposed method is not as accurate as the MTTF simulation, its tremendous speedup values are very compelling for early system reliability assessment. In fact, to perform the MTTF simulation, we need to obtain a complete design and verification test which may take several months of development. If the design cannot pass the reliability requirements, the waste in re-design time and resources can be extremely critical.

7. Conclusion

This chapter presented a detailed faults/defects analysis and an efficient reliability assessment method to approximate the lifetime reliability of a NoC system and compares it with a system-level simulation. In addition, this chapter presented an architecture and design of a fault-tolerant TSV-NoC system which can handle major failures that can occur in TSV-based 3D-NoC systems.

With the aid of efficient mechanisms and algorithms, the presented TSV-NoC (3D-NoC) system is capable of detecting and recovering from soft errors which occur in the routing pipeline stages and leverages reconfigurable components to handle permanent faults in links, input buffers, and crossbars.

Through quantitative evaluation, we showed that the proposed system was able to recover efficiently from a significant number of soft and hard errors at different fault-rates, reaching up to 33% with a minimum of 96% successful packet arrival rate.

References

[1] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, “3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration,” *Proc. IEEE*, vol. 89, no. 5, pp. 602–633, 2001.

[2] B. Swinnen *et al.*, “3D integration by Cu-Cu thermo-compression bonding of extremely thinned bulk-Si die containing 10 μm pitch through-Si vias,” in *Int. Electron Devices Meeting*, pp. 1–4, IEEE, 2006.

[3] A. B. Abdallah and M. Sowa, “Basic Network-on-Chip Interconnection for Future Gigascale MCSoCs Applications: Communication and Computation Orthogonalization,” in *Proc. of the Symposium on Science, Society, and Technology (TJASSST2006)*, pp. 1–7, 2006.

[4] A. Ben Ahmed *et al.*, “Architecture and Design of Efficient 3D Network-on-Chip (3D NoC) for Custom Multicore SoC,” in *2010 Int. Conf. on Broadband, Wireless Computing, Communication and Applications*, pp. 67–73, Nov 2010.

[5] A. B. Ahmed, A. B. Abdallah, and K. Kuroda, “Architecture and design of efficient 3d network-on-chip (3d noc) for custom multicore soc,” in *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, pp. 67–73, IEEE, 2010.

[6] A. B. Ahmed, Y. Okuyama, and A. B. Abdallah, “Non-blocking electro-optic network-on-chip router for high-throughput and low-power many-core systems,” in *Information Technology and Computer Applications Congress (WCITCA), 2015 World Congress on*, pp. 1–7, IEEE, 2015.

[7] B. A. Abderazek, M. Masuda, A. Canedo, and K. Kuroda, “Natural instruction level parallelism-aware compiler for high-performance queuecore processor architecture,” *The Journal of supercomputing*, vol. 57, no. 3, pp. 314–338, 2011.

[8] D. Fick *et al.*, “A highly resilient routing algorithm for fault-tolerant NoCs,” in *2009 Design, Automation Test in Europe Conf. Exhibition*, pp. 21–26, Apr 2009.

[9] E. Karl *et al.*, “Reliability Modeling and Management in Dynamic Microprocessor-based Systems,” in *Proceedings of the 43rd Annu. Design Automation Conf.*, DAC ’06, (New York), pp. 1057–1060, ACM, 2006.

[10] ITRS, “2012 Edition Update Process Integration, Devices, and Structures,” tech. rep., The Int. Technology Roadmap for Semiconductor, 2012. `http://www.itrs2.net/2012-itrs.html`(accessed 16.06.16).

[11] F. Ye and K. Chakrabarty, “TSV open defects in 3D integrated circuits: Characterization, test, and optimal spare allocation,” in *Proc. 49th Annual Design Automation Conf.*, pp. 1024–1030, ACM, 2012.

[12] J. Zhao *et al.*, “Overview of 3D Architecture Design Opportunities and Techniques,” *IEEE Des. Test.*, vol. PP, pp. 2168–2356, Jul 2015.

[13] U. Kang *et al.*, “8Gb 3D DDR3 DRAM using through-silicon-via technology,” in *IEEE Int. Solid-State Circuits Conf.-Dig. of Tech. Papers*, pp. 130–131, IEEE, 2009.

[14] M. Radetzki *et al.*, “Methods for fault tolerance in networks-on-chip,” *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 8, 2013.

[15] K. Constantinides *et al.*, “Bulletproof: A defect-tolerant CMP switch architecture,” in *The Twelfth Int. Symp. on High-Performance Computer Architecture*, pp. 5–16, IEEE, 2006.

[16] X. Ni, E. Meneses, N. Jain, and L. V. Kalé, “Acr: Automatic checkpoint/restart for soft and hard error protection,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC ’13, pp. 7:1–7:12, ACM, 2013.

[17] K. N. Dang, M. Meyer, Y. Okuyama, and A. B. Abdallah, “A low-overhead soft–hard fault-tolerant architecture, design and management scheme for reliable high-performance many-core 3D-NoC systems,” *The Journal of Supercomputing*, pp. 1–25, 2017.

[18] A. B. Ahmed and A. B. Abdallah, “Adaptive fault-tolerant architecture and routing algorithm for reliable many-core 3D-NoC systems,” *Journal of Parallel and Distributed Computing*, vol. 93-94, pp. 30–43, 2016.

[19] K. N. Dang, A. B. Ahmed, Y. Okuyama, and A. B. Abdallah, “Scalable design methodology and online algorithm for TSV-cluster defects recovery in highly reliable 3D-NoC Systems,” *IEEE Transactions on Emerging Topics in Computing*, 2017 (in press).

[20] S. Lin *et al.*, “Automatic-repeat-request error-control schemes,” *IEEE Communications Magazine*, vol. 22, no. 12, pp. 5–17, 1984.

[21] D. Bertozzi *et al.*, “Error control schemes for on-chip communication links: the energy-reliability tradeoff,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, vol. 24, pp. 818–831, Jun 2005.

[22] Q. Yu and P. Ampadu, “Transient and permanent error co-management method for reliable networks-on-chip,” in *Fourth ACM/IEEE Int. Symp. on Networks-on-Chip (NOCS)*, pp. 145–154, IEEE, 2010.

[23] D. Ernst *et al.*, “Razor: A low-power pipeline based on circuit-level timing speculation,” in *Proceedings 36th Annu. IEEE/ACM Int. Symp. on Microarchitecture (MICRO-36)*, pp. 7–18, IEEE, 2003.

[24] K. N. Dang, M. Meyer, Y. Okuyama, X.-T. Tran, and A. Ben Abdallah, “A Soft-Error Resilient 3D Network-on-Chip Router,” in *IEEE 7th International Conference on Awareness Science and Technology (iCAST)*, pp. 84–90, September 2015.

[25] Q. Yu *et al.*, “Addressing network-on-chip router transient errors with inherent information redundancy,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, pp. 105:1–105:21, Jul 2013.

[26] A. Prodromou *et al.*, “NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures,” in *Proceedings of the 2012 45th Annu. IEEE/ACM Int. Symp. on Microarchitecture (MICRO)*, pp. 60–71, Dec 2012.

[27] R. Parikh and V. Bertacco, “Formally Enhanced Runtime Verification to Ensure NoC Functional Correctness,” in *Proceedings of the 44th Annu. IEEE/ACM Int. Symp. on Microarchitecture*, MICRO-44, (New York), pp. 410–419, ACM, 2011.

[28] A. B. Ahmed, T. Ochi, S. Miura, and A. B. Abdallah, “Run-time monitoring mechanism for efficient design of application-specific noc architectures in multi/manycore era,” in *Complex, Intelligent, and Software Intensive Systems (CISIS), 2013 Seventh International Conference on*, pp. 440–445, IEEE, 2013.

[29] T. Lehtonen *et al.*, "Self-adaptive system for addressing permanent errors in on-chip interconnects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 4, pp. 527–540, 2010.

[30] S. Shamshiri and K.-T. Cheng, "Yield and cost analysis of a reliable NoC," in *27th IEEE VLSI Test Symp. (VTS'09)*, pp. 173–178, IEEE, 2009.

[31] C. Hernández *et al.*, "Dealing with variability in NoC links," in *2nd Workshop on Diagnostic Services in Network-on-Chips*, pp. 4–10, Jun 2008.

[32] A. Ben Ahmed and A. Ben Abdallah, "Architecture and design of high-throughput, low-latency, and fault-tolerant routing algorithm for 3D-network-on-chip (3D-NoC)," *The Journal of Supercomputing*, vol. 66, no. 3, pp. 1507–1532, 2013.

[33] A. DeOrio *et al.*, "A reliable routing architecture and algorithm for NoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 5, pp. 726–739, 2012.

[34] A. Ben Ahmed and A. Ben Abdallah, "Graceful deadlock-free fault-tolerant routing algorithm for 3D Network-on-Chip architectures," *Journal of Parallel and Distributed Computing*, vol. 74, no. 4, pp. 2229–2240, 2014.

[35] A. B. Ahmed, M. Meyer, Y. Okuyama, and A. B. Abdallah, "Adaptive error-and traffic-aware router architecture for 3d network-on-chip systems," in *Embedded Multicore/Manycore SoCs (MCSoc), 2014 IEEE 8th International Symposium on*, pp. 197–204, IEEE, 2014.

[36] A. Ben Ahmed *et al.*, "Deadlock-Recovery Support for Fault-tolerant Routing Algorithms in 3D-NoC Architectures," in *2013 IEEE 7th Int. Symp. on Embedded Multicore Socs*, pp. 67–72, Sep 2013.

[37] Y. Zhao *et al.*, "Cost-effective TSV grouping for yield improvement of 3D-ICs," in *Asian Test Symp.*, pp. 201–206, IEEE, 2011.

[38] I. Loi *et al.*, "A low-overhead fault tolerance scheme for TSV-based 3D network on chip links," in *Proc. 2008 IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 598–602, 2008.

[39] L. Jiang, Q. Xu, and B. Eklow, "On effective through-silicon via repair for 3-D-stacked ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, vol. 32, no. 4, pp. 559–571, 2013.

[40] T. Zhang *et al.*, "Temperature-aware routing in 3D ICs," in *Asia and South Pacific Conf. on Design Automation*, pp. 309–314, Jan 2006.

[41] M. Cho *et al.*, "Design method and test structure to characterize and repair TSV defect induced signal degradation in 3D system," in *Proc. Int. Conf. on Computer-Aided Design*, pp. 694–697, 2010.

[42] J. U. Knickerbocker *et al.*, "Three-dimensional silicon integration," *IBM J. Research and Development*, vol. 52, no. 6, pp. 553–569, 2008.

[43] Y. Zhao *et al.*, "Online Fault Tolerance Technique for TSV-Based 3-D-IC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 23, no. 8, pp. 1567–1571, 2015.

[44] A.-C. Hsieh and T. Hwang, "TSV redundancy: architecture and design issues in 3-D IC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 20, no. 4, pp. 711–722, 2012.

[45] A. Eghbal *et al.*, "Analytical Fault Tolerance Assessment and Metrics for TSV-based 3D Network-on-Chip," *IEEE Trans. Comput.*, vol. 64, pp. 3591–3604, Dec 2015.

[46] G. Van der Plas *et al.*, "Design issues and considerations for low-cost 3-D TSV IC technology," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 293–307, 2011.

[47] United States of America: Department of Defense, *Military Handbook: Reliability Prediction of Electronic Equipment: MIL-HDBK-217F*. 1991.

[48] M. Zhang and N. R. Shanbhag, "Soft-error-rate-analysis (SERA) methodology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2140–2155, 2006.

[49] H. T. Nguyen, Y. Yagil, N. Seifert, and M. Reitsma, "Chip-level soft error estimation method," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 365–381, 2005.

[50] IEEE Standards Coordinating Committee, *IEEE Guide for Selecting and Using Reliability Predictions. IEEE 1413.1*. IEEE, 2002.

[51] A. Simevski, R. Kraemer, and M. Krstic, "Automated integration of fault injection into the ASIC design flow," in *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, pp. 255–260, IEEE, 2013.

[52] M. L. Shooman, *Reliability of computer systems and networks: fault tolerance, analysis, and design*. John Wiley & Sons, 2003.

[53] A. Dalirsani, M. Hosseinabady, and Z. Navabi, "An analytical model for reliability evaluation of NoC architectures," in *13th IEEE International On-Line Testing Symposium (IOLTS 2007)*, pp. 49–56, IEEE, 2007.

[54] J. B. Bernstein, M. Gurfinkel, X. Li, J. Walters, Y. Shapira, and M. Talmor, "Electronic circuit reliability modeling," *Microelectronics Reliability*, vol. 46, no. 12, pp. 1957–1979, 2006.

[55] A. Y. Yamamoto and C. Ababei, "Unified reliability estimation and management of NoC based chip multiprocessors," *Microprocessors and Microsystems*, vol. 38, no. 1, pp. 53–63, 2014.

[56] K. Aisopos, C.-H. O. Chen, and L.-S. Peh, "Enabling system-level modeling of variation-induced faults in networks-on-chips," in *Proceedings of the 48th Design Automation Conference*, pp. 930–935, ACM, 2011.

[57] D. Crowe and A. Feinberg, *Design for reliability*, vol. 11. CRC press, 2001.

[58] R. Cressent, P. David, V. Idasiak, and F. Kratz, "Designing the database for a reliability aware Model-Based System Engineering process," *Reliability Engineering & System Safety*, vol. 111, pp. 171–182, 2013.

[59] J. A. Rivers, M. S. Gupta, J. Shin, P. N. Kudva, and P. Bose, "Error tolerance in server class processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 30, pp. 945–959, 2011.

[60] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebalis, "A reliability-aware design methodology for networks-on-chip applications," in *Design & Technology of Integrated Systems in Nanoscal Era, 2009. DTIS'09. 4th International Conference on*, pp. 107–112, IEEE, 2009.

[61] L. Huang, F. Yuan, and Q. Xu, "Lifetime reliability-aware task allocation and scheduling for MPSoC platforms," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 51–56, European Design and Automation Association, 2009.

[62] J. B. Bowles, "A survey of reliability-prediction procedures for microelectronic devices," *IEEE Transactions on Reliability*, vol. 41, no. 1, pp. 2–12, 1992.

[63] J. Shin, V. Zyuban, Z. Hu, J. A. Rivers, and P. Bose, "A framework for architecture-level lifetime reliability modeling," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pp. 534–543, IEEE, 2007.

[64] T. Lehtonen, P. Liljeberg, and J. Plosila, "Fault tolerance analysis of NoC architectures," in *2007 IEEE International Symposium on Circuits and Systems*, pp. 361–364, IEEE, 2007.

[65] H. Zimmer and A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip," in *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pp. 188–193, ACM, 2003.

[66] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. R. Das, "Design and analysis of an noc architecture from performance, reliability and energy perspective," in *Proceedings of the 2005 ACM symposium on Architecture for networking and communications systems*, pp. 173–182, ACM, 2005.

[67] A. Savino, S. Di Carlo, G. Politano, A. Benso, A. Bosio, and G. Di Natale, "Statistical reliability estimation of microprocessor-based systems," *IEEE Transactions on Computers*, vol. 61, no. 11, pp. 1521–1534, 2012.

[68] M. Khayambashi *et al.*, "Analytical reliability analysis of 3D NoC under TSV failure," *ACM J. Emerging Technologies in Computing Systems*, vol. 11, no. 4, p. 43, 2015.

[69] K. N. Dang, A. B. Ahmed, X.-T. Tran, Y. Okuyama, and A. B. Abdallah, "A comprehensive reliability assessment of fault-resilient network-on-chip using analytical model," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 11, pp. 3099–3112, 2017.

[70] A. Bobbio and K. S. Trivedi, "An Aggregation Technique for the Transient Analysis of Stiff Markov Chains," *IEEE Transactions on Computers*, vol. C-35, pp. 803–814, September 1986.

[71] J. A. Cheatham, J. M. Emmert, and S. Baumgart, "A survey of fault tolerant methodologies for FPGAs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 11, no. 2, pp. 501–533, 2006.

[72] K. N. Dang, M. Meyer, Y. Okuyama, and A. B. Abdallah, "Reliability Assessment and Quantitative Evaluation of Soft-Error Resilient 3D Network-on-Chip Systems," in *IEEE 25th Asian Test Symposium*, pp. 161–166, IEEE, 2016.

[73] J. P. Hayes, "A graph model for fault-tolerant computing systems," *IEEE Transactions on Computers*, vol. 100, no. 9, pp. 875–884, 1976.

[74] K. N. Dang, Y. Okuyama, and A. B. Abdallah1, "Soft-error resilient Network-on-Chip for safety-critical applications," in *2016 International Conference on IC Design and Technology (ICICDT)*, pp. 1–4, June 2016.

[75] M.-Y. Hsiao, "A class of optimal minimum odd-weight-column sec-ded codes," *IBM Journal of Research and Development*, vol. 14, no. 4, pp. 395–401, 1970.

[76] A. B. Ahmed and A. B. Abdallah, "Low-overhead Routing Algorithm for 3D Network-on-Chip," in *2012 Third International Conference on Networking and Computing (ICNC)*, pp. 23–32, December 2012.

[77] M. Laisne *et al.*, "Systems and methods utilizing redundancy in semiconductor chip interconnects," 2013. US Patent 8,384,417.

[78] M. Tsai *et al.*, "Through silicon via (TSV) defect/pinhole self test circuit for 3D-IC," in *IEEE Int. Conf. on 3D System Integration*, pp. 1–8, IEEE, 2009.

[79] Y.-J. Huang and J.-F. Li, "Built-in self-repair scheme for the TSVs in 3-D ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 10, pp. 1600–1613, 2012.

[80] Z. Qian and C. Y. Tsui, "A thermal-aware application specific routing algorithm for Network-on-Chip design," in *16th Asia and South Pacific Design Automation Conf.*, pp. 449–454, Jan 2011.

[81] M. Palesi *et al.*, "Application specific routing algorithms for networks on chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, pp. 316–330, Mar 2009.

[82] Y. Ghidini *et al.*, “Lasio 3D NoC vertical links serialization: Evaluation of latency and buffer occupancy,” in *26th Symp. on Integrated Circuits and Systems Design*, pp. 1–6, Sep 2013.

[83] A. A. Chien and J. H. Kim, “Planar-adaptive routing: low-cost adaptive networks for multiprocessors,” *Journal of the ACM*, vol. 42, no. 1, pp. 91–123, 1995.

[84] R. Sivaram, “Queuing delays for uniform and nonuniform traffic patterns in a MIN,” *ACM SIGSIM Simulation Dig.*, vol. 22, no. 1, pp. 17–27, 1992.

[85] P. Chen *et al.*, “The parallel algorithm implementation of matrix multiplication based on ESCA,” in *IEEE Asia Pacific Conf. on Circuits and Systems*, pp. 1091–1094, IEEE, 2010.

[86] A. S. Zekri and S. G. Sedukhin, “The general matrix multiply-add operation on 2D torus,” in *20th Int. Parallel and Distributed Processing Symp.*, pp. 8–16, IEEE, 2006.

[87] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[88] A.-M. Rahmani *et al.*, “High-performance and fault-tolerant 3D noc-bus hybrid architecture using arb-net-based adaptive monitoring platform,” *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 734–747, 2014.

[89] D. Bertozzi *et al.*, “NoC synthesis flow for customized domain specific multiprocessor systems-on-chip,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 113–129, 2005.

[90] A. Ben Ahmed and A. Ben Abdallah, “LA-XYZ: low latency, high throughput look-ahead routing algorithm for 3D network-on-chip (3D-NoC) architecture,” in *IEEE 6th International Symposium on Embedded Multicore Socs (MCSoC)*, pp. 167–174, IEEE, September 2012.

MA